Fachhochschule Köln

University of Applied Sciences Cologne

Campus Gummersbach

Fakultät für Informatik und Ingenieurwissenschaften

Studiengang Allgemeine Informatik

Diplomarbeit

zur Erlangung

des Diplomgrades

Diplom-Informatiker (FH)

in der Fachrichtung Informatik

Entwicklung einer Club/Disco-Verwaltung auf Basis einer jsp/MySql-Applikation

Erstprüferin:	Prof. Dr. Heide Faeskorn-Woyke
Zweitprüfer:	Prof. Dr. Holger Günther
vorgelegt am:	9.8.2004
von cand.:	Stefan Höpp
aus:	Hohe Str. 24
	50667 Köln
E-Mail:	Stefan@Hoepp.de
Mat.-Nr.:	11021117

Inhalt

Abbildungsverzeichnis 7
Tabellenverzeichnis 8
Abkürzungs- und Symbolverzeichnis 11
1 Einleitung 12
1.1 Anmerkungen zum Sprachgebrauch 12
1.2 Anmerkungen zu Listings 12
1.3 Übersicht 12
2 Ist-Zustand 15
2.1 Mitarbeitermanagement 15
2.2 Kassensystem 16
2.3 Deejaying 17
2.4 Abendbilanz 18
2.5 Zusammenfassung 19
3 Ziele 22
3.1 Mitarbeitermanagement 22
3.2 Kassensystem 23
3.3 Deejaying 23
3.4 Abendbilanz 24
3.5 Zusammenfassung 24
4 Pflichtenheft 26
4.1 Zielbestimmung 26
4.2 Produkteinsatz 28
4.3 Produktumgebung 29
4.4 Produktfunktionen 30
4.5 Produktdaten 31
4.6 Produktleistungen 31
4.7 Nutzeroberfläche 31
4.8 Qualitätszielbestimmung 31
4.9 Globale Testfälle 32
4.10 Entwicklungsumgebung 32
5 Grundsätzliches 33
5.1 Visualisierung 33
5.2 Tomcat 37

5.2.1 Webapplikation 37
5.2.2 Benutzermanagement 38
5.3 Reguläre Ausdrücke 41
5.4 Mensch-Computer-Interaktion 42
5.4.1 Ziele 42
5.4.2 Eingabe von Informationen 43
5.4.3 Darstellung von Informationen 44
5.4.4 Navigation 46
5.5 Informationshaltung 47
5.5.1 Datenbank 47
5.5.2 Import/Export 48
6 Architektur 49
6.1 Horizontal 49
6.2 Vertikal 50
6.3 Modulfunktionsübersicht 53
6.3.1 Modul Abrechnung 53
6.3.2 Modul Mitarbeiterverwaltung 53
6.3.3 Modul Personalmanagement 54
6.3.4 Modul Kassensystem 54
6.3.5 Modul Deejaying 54
6.3.6 Modul Administration 55
7 Informationshaltung 56
7.1 Datenbank 56
7.1.1 Grundsätzliches 56
7.1.2 mysql-connector 57
7.1.3 Anfragen parsen 60
7.1.4 Datenbank administrieren 62
7.2 Files schreiben und lesen 64
7.3 Import/Export 66
8 Programmstart 68
9 Modul Mitarbeiterverwaltung 69
9.1 Anzeigen 69
9.2 Anlegen 71
9.3 Ändern 72

9.4 Löschen 73
10 Modul Mitarbeitermanagement 74
10.1 Anzeigen 74
10.2 Buchen 75
10.3 Löschen 76
10.4 Vorbelegung anlegen 76
10.5 Vorbelegung anzeigen 77
10.6 Vorbelegung löschen 77
11 Modul Abrechnung 78
11.1 Grundsätzliches 78
11.2 Lohnübersicht 79
11.3 Abend anzeigen 80
11.4 Auszahlungen 81
11.5 Tabellarisch 82
11.6 Grafisch 82
11.6.1 jFreeChart 83
11.7 Freigabe 85
11.7.1 Tag freigeben 86
11.7.2 Monat freigeben 89
11.7.3 Tag Unfreigabe 90
12 Modul Kassensystem 92
12.1 Locking 92
13 Modul Deejaying 97
13.1 Playlist 97
13.1.1 Hinzufügen 97
13.1.2 Anzeigen 98
13.2 Wunschliste 99
13.2.1 Hinzufügen 99
13.2.2 Anzeigen 99
13.3 Archiv 99
13.3.1 Import 100
13.3.2 Export 100
14 Modul Administration 101
14.1 Artikel 101

14.1.1 Anlegen 101
14.1.2 Anzeigen 102
14.1.3 Löschen 102
14.2 Theke 102
14.2.1 Anlegen 102
14.2.2 Anzeigen 103
14.2.3 Löschen 103
14.3 Kasse 103
14.3.1 Anlegen 103
14.3.2 Anzeigen 103
14.3.3 Löschen 104
14.4 Vertrag 104
14.4.1 Anlegen 104
14.4.2 Anzeigen 104
14.4.3 Löschen 104
14.4.4 Steuer anlegen 105
14.4.5 Steuer anzeigen 105
14.4.6 Steuer löschen 105
15 Implementierungsdetails 106
15.1 Package-Hierarchie 106
15.2 de.hoepp.date 108
15.2.1 CalendarDate 108
15.2.2 Period 111
15.2.3 TimePeriod 112
15.3 Schutz vor falschen Eingaben 112
15.3.1 Senden 112
15.3.2 Empfangen 114
16 Zusammenfassung und Ausblick 115
Literaturverzeichnis 117
Print-Literatur 117
Online-Literatur 118
Anhang A Installation 119
A-1 MySql 119
A-2 MySql-Connector 119

A-3 MySqlAdministrator 119
A-4 MySqlControlCenter 119
A-5 JFreeChart 120
A-6 Tomcat 120
Anhang B ER-Diagramme 121
Anhang C UML-Diagramme 123
Anhang D Seitenverlauf 128
Anhang E Reguläre Ausdrücke 131
Anhang F Erklärung 135

Abbildungsverzeichnis

Abbildung 2-1 Vorbelegung eines Mitarbeiters 15
Abbildung 2-2 Buchung eines Mitarbeiters 16
Abbildung 2-3 Abrechnung eines Mitarbeiters 16
Abbildung 2-4 Artikel kaufen 17
Abbildung 2-5 Eintrittskarte kaufen 17
Abbildung 2-6 Garderobe abgeben 17
Abbildung 2-7 Titel/Interpret suchen 18
Abbildung 2-8 Abendbilanz erstellen 18
Abbildung 2-9 Ist-Zustand Abendabrechnung 20
Abbildung 3-1 einheitliches Mitarbeitermanagement 22
Abbildung 3-2 einheitliches Kassensystem 23
Abbildung 3-3 Interpret/Titel suchen 24
Abbildung 3-4 vereinfachte Abendbilanz 24
Abbildung 3-5 vereinfachte Abendabrechnung 25
Abbildung 4-5-1 statische Seiten 33
Abbildung 4-5-2 Generierung einer dynamischen Seite 34
Abbildung 5-3 Ein Beispiel für ein einfaches Servlet 35
Abbildung 5-4 Ausliefern einer jsp-Seite 36
Abbildung 5-5 Beispiel eine jsp-Seite 37
Abbildung 5-6 Ordnerstruktur Webapplikation 38
Abbildung 5-7 Tomcat 5.0 Admin-Tool 39
Abbildung 5-8 Tomcat-users.xml 40
Abbildung 5-9 Beispiel für eine server.xml 41
Abbildung 5-10 schlechte Darstellung der Informationen 44
Abbildung 5-11 gute Darstellung der Informationen 45
Abbildung 5-12 Schaltflächen sinngemäß beschriften 46
Abbildung 5-13 Baumnavigation 46
Abbildung 5-14 Beispiel einer csv-Datei 48
Abbildung 6-1 Horizontaler Aufbau der Applikation 49
Abbildung 6-2 Vertikaler Aufbau der Applikation 51
Abbildung 7-1 Zugriff auf Datenbank über msql-connector 57
Abbildung 7-2 Herstellen einer Verbindung mit dem mysql-connector 58

Abbildung 7-3 Erzeugen eines SQL-Statements 58
Abbildung 7-4 Erzeugen eines executeUpdate 59
Abbildung 7-5 Beispiel für die Benutzung eines ResultSet 59
Abbildung 7-6 Parsen eines Strings 60
Abbildung 7-7 Eingabe in ein Text-Input-Feld 61
Abbildung 7-8 ungefährliche Formulareingabe 61
Abbildung 7-9 gefährliche Formulareingabe 62
Abbildung 7-10 SQL-Injection Schwachstelle 62
Abbildung 7-11 Das MySQLControlCenter 63
Abbildung 7-12 Der MySQLAdministrator 64
Abbildung 7-13 FileDataHandler kapselt den Dateizugriff 64
Abbildung 7-14 In eine Datei schreiben 66
Abbildung 7-15 Aus einer Datei lesen 66
Abbildung 7-16 Import/Export aus der Datenbank 67
Abbildung 8-1 Programmstart 68
Abbildung 9-1 Mitarbeiterliste 69
Abbildung 9-2 Detailansicht eines Mitarbeiters 70
Abbildung 9-3 Anlegen eines Mitarbeiters 72
Abbildung 10-1 Erstellung des Schichtplans 74
Abbildung 10-2 Den Personalplan anzeigen 74
Abbildung 10-3 Einen Mitarbeiter buchen 75
Abbildung 10-4 Einen Tag vorbelegen 76
Abbildung 11-1 Lohnübersicht 80
Abbildung 11-2 Abendabrechnung anzeigen 81
Abbildung 11-3 Ausgabe der zu tätigenden Überweisungen 82
Abbildung 11-4 Grafische Auswertung der Umsätze und Bilanzen 83
Abbildung 11-5 Code-Beispiel eines Liniendiagramms 85
Abbildung 11-6 Tag und Monat freigeben 86
Abbildung 11-7 Einen Tag freigeben 87
Abbildung 11-8 Einen Monat freigeben 89
Abbildung 11-9 Einen Tag unfreigeben 91
Abbildung 12-1 Ablauf Kassensystem 92
Abbildung 12-2 Anmeldung Kassensystem 93
Abbildung 12-3 Locking-Mechanismus 93

Abbildung 12-4 Das Kassensystem 94
Abbildung 13-1 Titel einer Playlist hinzufügen 97
Abbildung 13-2 Playlist anzeigen 98
Abbildung 15-1 Die Package Hierarchie 106
Abbildung 15-2 Veranschaulichung der Klassengeflechts 108
Abbildung 15-3 ungünstiges Anlegen zweier identischer Daten 110
Abbildung 15-4 Zusammenbauen einer Datum String Repräsentation 110
Abbildung 15-5 Datum in Datenbank schreiben und daraus lesen 111
Abbildung 15-6 Script zur Validierung der Eingaben 113

Tabellenverzeichnis

Tabelle 1 Table Vertrag 78
Tabelle 2 Table Besteuerung 78
Tabelle 3 Table Steuern 79
Tabelle 4 Parameter beim Anlegen eines Chart 84
Tabelle 5 Datumsformate bei Charts 84
Tabelle 6 Table Booking 87
Tabelle 7 Table Arbeitskonto 88
Tabelle 8 Table Tageszahlen 88
Tabelle 9 Table Einnahmen 88
Tabelle 10 Table Einnahmenkonto 88
Tabelle 11 Table Tageszahlen 88
Tabelle 12 Table Monatsauszahlungen 90
Tabelle 13 Table Monatseinnahmen 90
Tabelle 14 Table Monatsauszahlungen 90
Tabelle 15 Table Playlist 98
Tabelle 16 Table CDArchiv 100
Tabelle 17 Table Artikel 101

Abkürzungs- und Symbolverzeichnis

ASCII.....American Standard Code for Information Interchange

API.....Application Programming Interface

CGI ..... Common Gateway Interface

CMS ..... Content Management System

CRT..... Cathode-Ray Tube

CSV ..... Comma Separated Values

DBS ..... Datenbanksystem

GEMA.....Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte

HTML..... Hypertext Markup Language

HTTP .....Hypertext Transfer Protocol

IEEE..... Institute of Electrical and Electronics Engineers

JDBC ..... Java Database Connectivity

JDK.....Java Development Kit

JSP .....Java Server Pages

JVM .....Java Virtual Maschine

LAN .....Local Area Network

MCI.....Mensch-Computer-Interaktion

QWERTZ..... die obere Reihe in einem genormten Tastatur Layout

SQL..... Structured Query Language

1 Einleitung

Kein Club/Disco-Betrieb kommt heute mehr ohne eine mehr oder weniger gut ausgebaute IT-Infrastruktur aus. Das fängt an bei kleinen Clubs, welche über autonome Kassensysteme und ein Abrechnungssystem verfügen, und kennt nach oben hin keine Grenze. So verwalten große Clubs nicht nur Abrechnung und die Kassen, sondern verfügen auch über autonome Systeme, welche beispielsweise die Licht-, beziehungsweise Audioanlage steuern. Allen diesen Systemen ist gemein, dass sie bis auf wenige individuell programmierte Software keine Lösung für alles bieten.

Diese Diplomarbeit entstand aus der Motivation, diesem Abhilfe zu schaffen. Das heißt, mit dieser Diplomarbeit soll der Grundstein gelegt werden für eine Software, welche diesen Missstand behebt. So wird dieses Produkt die verschiedenen Anwendungsgebiete in einer Applikation vereinen.

1.1 Anmerkungen zum Sprachgebrauch

Die Begriffe Club und Disco werden hier synonym verwendet. Bei beiden Begrifflichkeiten handelt es sich um eine Abkürzung für einen Club/Disco-Betrieb.

1.2 Anmerkungen zu Listings

Die Code-Beispiele sind, wenn nicht ausdrücklich anders erwähnt, nur auszugsweise abgebildet und der Übersichtlichkeit halber um selbstverständliche Elemente, wie zum Beispiel Importe reduziert. Einrückungen ordnen sich teilweise ebenfalls der Übersichtlichkeit unter.

1.3 Übersicht

Einleitung
beschreibt die Motivation dieser Arbeit und macht einige Anmerkungen zum Aufbau und der Terminologie.

IST-Zustand

analysiert den momentanen Zustand und nimmt als Grundlage hierfür einen Querschnitt über diverse Clubs.

Ziele

defniert die Ziele dieser Diplomarbeit.

Pflichtenheft

ist die formale Repräsentation der Ziele der Diplomarbeit.

Grundsätzliches

erklärt grundsätzliche Technologien, welche für die Diplomarbeit verwendet wurden und deren Verständnis für das Lesen und Verstehen der Diplomarbeit nötig sind.

Architektur

beschäftigt sich mit der horizontalen und vertikalen Aufteilung der Applikation. Des weiteren wird hier noch eine Übersicht gegeben über die einzelnen Module der Applikation und ihre Funktionen.

Informationshaltung

beschreibt die Form der Informationshaltung. Es wird erklärt, welche Datenbank ausgewählt wurde, und warum. Zudem beschäftigt es sich mit dem Thema Import und Export von Daten.

Programmstart

skizziert einige der zahlreichen Einstellungen und Initialisierungen, die beim Programmstart vorgenommen werden.

Modul Mitarbeiterverwaltung

beschreibt das Modul, welches sich mit der Verwaltung der Mitarbeiter beschäftigt. Dieses Modul ist ausschließlich zur Verwaltung der Mitarbeiterkerndaten bestimmt und beinhaltet keinerlei Abrechnungs- oder Buchungsfunktionalität.

Modul Mitarbeitermanagement

erklärt das Modul, welches sich mit allen Funktionen rund um das Buchen von Mitarbeitern beschäftigt. Hier wird erklärt, was der Unterschied zwischen Buchen und Vorbelegung ist und wie das Freigabesystem arbeitet und benutzt wird.

Modul Abrechnung

beschäftigt sich mit dem Modul, welches sich um alle Belange der Abrechnung kümmert. Das fängt an bei der Bildung der Abendbilanzen, geht über die tabellarische und grafische Auswertung von Abrechnungszeiträumen und endet bei der Ausgabe der monatlich vorzunehmenden Überweisungen von Löhnen und Steuern.

Modul Kassensystem

beschreibt das Modul, dass das Kassensystem darstellt, mit Hilfe dessen man Artikel buchen kann.

Modul Administration

erklärt das Modul, mit welchem die Grundeinstellungen der Applikation eingesehen und geändert werden können. Hier können beispielsweise neue Vertragsarten und Besteuerungen usw. angelegt oder gelöscht werden.

Modul Implementierungsdetails

beschäftigt sich mit erwähnenswerten Implementierungsdetails der Applikation.

2 Ist-Zustand

Exemplarisch sei hier für die einzelnen Arbeitsbereiche (Kassenabrechnung, Personalmanagement, Abendbilanz und Deejaying) erklärt, wie dies aktuell betrieben wird. Dabei fällt schnell auf, dass für jeden Arbeitsschritt nicht unmittelbar beteiligte Arbeitsmittel notwendig sind. Es ist zum Beispiel für die Vorbelegung eines Mitarbeiters für einen Tag notwendig, vorher aus der Mitabeiterliste sein Kürzel herauszusuchen, um damit eine Vorbelegung in der dafür vorgesehenen Excel Tabelle machen zu können.

2.1 Mitarbeitermanagement

Das Mitarbeitermanagement teilt sich grob in drei Gruppen auf: Vorbelegung, Buchung und Abrechnung eines Mitarbeiters.

In der Regel bekommt jeder Mitarbeiter mindestens einen festen Tag zugewiesen, an dem er, neben den Vertretungsterminen, regelmäßig arbeitet. Für diesen Zweck gibt es eine Excel-Tabelle, in der die regelmäßigen Arbeitstage der Mitarbeiter verzeichnet sind, die so genannte Vorbelegungstabelle. Aus dieser Tabelle muss natürlich zum Ende des vorangegangenen Monats der Schichtplan für den kommenden Monat vorgefüllt werden. Es entsteht also aus dieser Tabelle das Grundgerüst des neuen Schichtplans (siehe Abbildung 2-1), welches später nur noch um die zusätzlich gebuchten Arbeitskräfte erweitert werden muss.

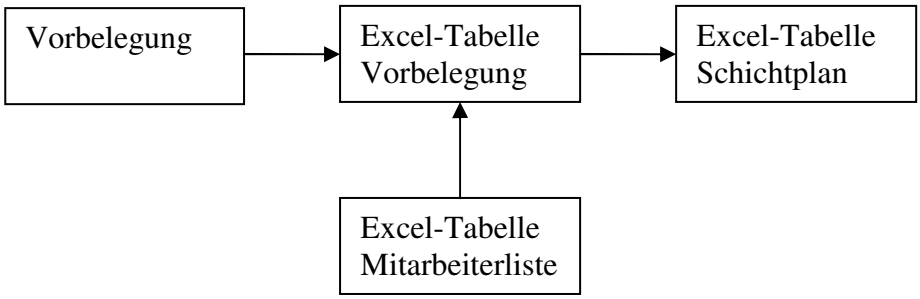


Abbildung 2-1 Vorbelegung eines Mitarbeiters

Falls nun an einem Tag ein Mitarbeiter arbeiten soll oder arbeitet, der an diesem Tag nicht als regulärer Mitarbeiter vorgesehen war, dann muss dieser natürlich für den Tag gebucht und in den Schichtplan eingetragen werden (siehe Abbildung

2-2). Gründe hierfür können sein, dass der reguläre Mitarbeiter zum Beispiel wegen Krankheit oder Urlaub ausfällt, oder aber, dass an diesem Tag kein regulärer Mitarbeiter eingeteilt ist.

Abbildung 2-2 Buchung eines Mitarbeiters

Sobald ein Abend beendet ist, das bedeutet, dass man nun weiß, welche Mitarbeiter an diesem Abend tatsächlich gearbeitet haben, kann man die Buchungen der Abrechnung übergeben. Hierfür ist es nun nötig, die Buchungen, die im Schichtplan eingetragen waren, als geleistete Stunden in das Datev[1] System einzugeben (siehe Abbildung 2-3).

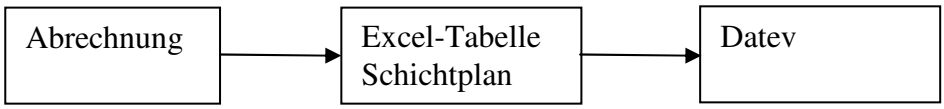


Abbildung 2-3 Abrechnung eines Mitarbeiters

2.2 Kassensystem

In jedem Club gibt es natürlich mehr als eine Kasse, welche entweder autonom, oder wie in diesem (besseren) Falle über ein Kassenrechnungssystem arbeiten. Dies hat den Vorteil, dass Änderungen, zum Beispiel an Preisen, hier nur an einer zentralen Stelle, nämlich dem Kassenrechnungssystem vorgenommen werden müssen. Trotzdem werden hier die unterschiedlichen Produkte und Leistungen, die verkauft werden, unterschiedlich abgewickelt.

Artikel, also im Wesentlichen Getränke, die man verkauft, werden über ein Kassensystem eingegeben und landen so im Kassenrechnungs-System (siehe Abbildung 2-4). Mit Hilfe dieses Systems wird am Ende des Abends die abgerechnete Summe mit dem tatsächlich in der Kasse vorhandenen Geldbetrag verglichen. Dieses System, welches für die Getränkeeinnahmen gut funktioniert,

[1] Datev ist ein System, das sich um alle anfallenden Abrechnungsbelange, also Steuern, Gehälter usw. kümmert.

wird jedoch bei den anderen Einnahmeformen durchbrochen und außer Kraft gesetzt.

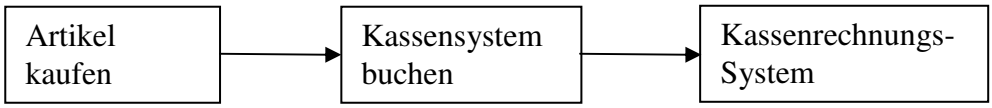


Abbildung 2-4 Artikel kaufen

In aller Regel werden die verkauften Eintrittskarten per Hand verzeichnet und dann für jeden Abend, getrennt als Summe der Einnahmen, in eine eigene Excel Tabelle eingetragen (siehe Abbildung 2-5).

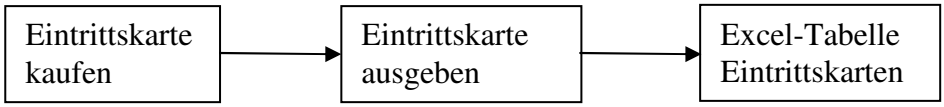


Abbildung 2-5 Eintrittskarte kaufen

Mit den Garderobenmarken verhält sich das ähnlich wie mit den Eintrittskarten. Am Ende des Abends wird anhand der fortlaufenden Nummer auf der letzten Garderobenmarke festgestellt, wie viele Garderobenmarken ausgegeben worden sind. Dies wird analog zur Eintrittskarten-Tabelle in eine Garderobeneinnahmen-Tabelle vermerkt (siehe Abbildung 2-6).

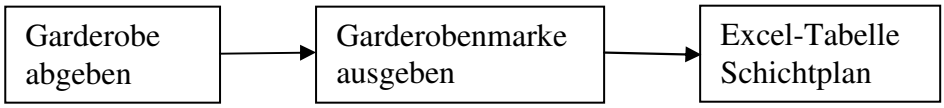


Abbildung 2-6 Garderobe abgeben

2.3 Deejaying

Dem Deejay steht in der Regel ein großes Plattenarchiv zur Verfügung. Während er auflegt[2], bedient er sich aus diesem. Damit er sich darin zurecht findet, stehen ihm eine Liste, häufig jedoch zwei Listen zur Verfügung (siehe Abbildung 2-7). Eine der beiden Listen ist nach Interpreten sortiert und eine nach Titeln. Dies ermöglicht eine Suche nach einer CD im Archiv, auch wenn man nur eine der beiden Information zur Verfügung hat. Hierfür ist es nötig (siehe Abbildung 2-7),

[2] mit Auflegen beschreibt man die Tätigkeit eines Deejays.

zwei Listen auf dem aktuellen Stand zu halten und wöchentlich auszudrucken, da neue Titel oder Interpreten alphabetisch einsortiert werden. Dies stellt auch einen gewissen finanziellen Aufwand dar, da beide Listen schnell auf über hundert Seiten anwachsen.

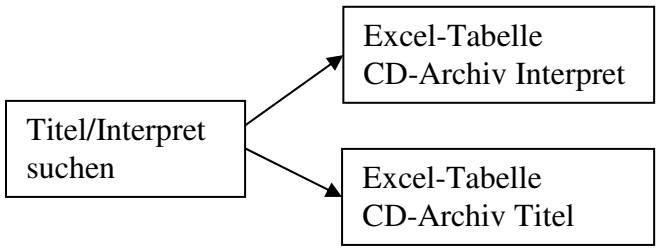


Abbildung 2-7 Titel/Interpret suchen

2.4 Abendbilanz

Bei der Erstellung der Abendbilanzen kommt die Hauptunzulänglichkeit der bisherigen Praxis am Deutlichsten zu tragen.

Alle Daten und Formeln, die benötigt werden, finden sich in separaten Systemen (siehe Abbildung 2-8), was eine einfache Bilanzermittlung unmöglich macht. Ebenso können Auswertungen der einzelnen Faktoren bestenfalls isoliert vorgenommen werden.

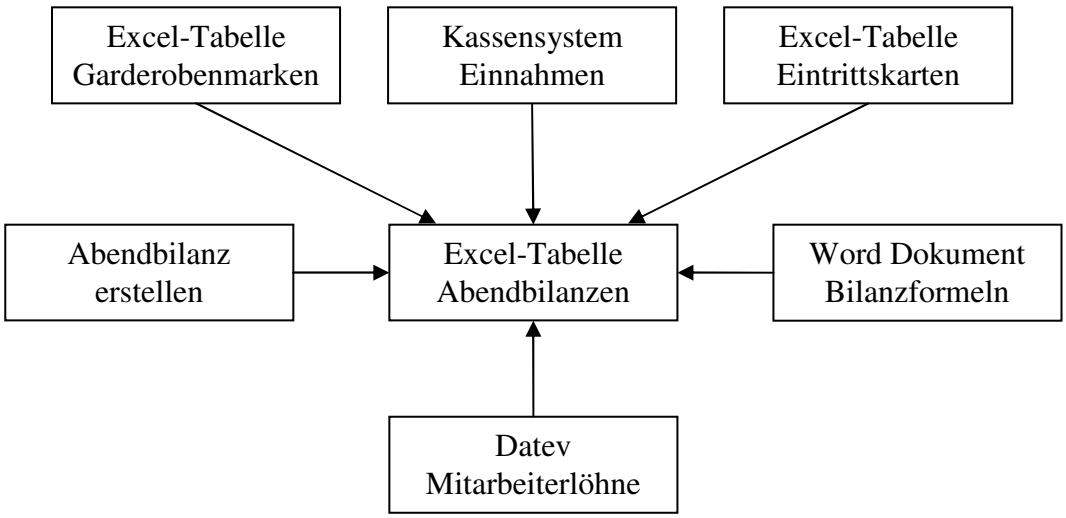


Abbildung 2-8 Abendbilanz erstellen

Um momentan eine Abrechung zu erstellen, braucht man die Informationen aus dem Kassensystem, der Eintrittskasse, der Garderobenkasse und dem Datev-System. Diese Daten müssen dann in ein ausgedrucktes Word-Dokument geschrieben werden. Darin befinden sich Formeln, nachdem in einzelnen Schritten von oben nach unten die Bilanz des Abends errechnet wird. Die Ergebnisse dieser Berechnung werden dann in eine Excel-Tabelle geschrieben, in der die Tagesergebnisse archiviert werden.

2.5 Zusammenfassung

Wenn man Abbildung 2-9 betrachtet, fallen direkt einige Unzulänglichkeiten und potentielle Fehlerquellen im Abendablauf ins Auge.

Zunächst fällt auf, dass Informationen mehrfach gehalten werden, die eigentlich zusammen gehören. So sind beispielsweise der Verkauf von Artikeln, Eintrittskarten oder Garderobenmarken von einander getrennt, obwohl es sich dabei im Grunde um dieselben Abläufe handelt. Dies erschwert auch die spätere Abrechnung, da die Ergebnisse dieser Verkäufe in unterschiedlichen Systemen landen und später zur Abrechnung erst wieder addiert werden und in eine neue Tabelle eingetragen werden müssen.

Wenn ein (neuer) Mitarbeiter einen festen Tag zugewiesen bekommt, dann müssen hier erst noch Werte aus einer anderen Tabelle (der Personalliste) entnommen und übertragen werden.

Sobald der Schichtplan für den kommenden Monat nun erstellt wird, müssen alle Werte aus der Vorbelegungstabelle per Hand in den Schichtplan übertragen werden. Da es sich bei diesem Schichtplan um eine Excel-Tabelle handelt, die keinerlei Verbindung zum Abrechnungssystem hat, sind die Buchungen sehr abstrakter Natur. Man kann lediglich sehen, dass ein Mitarbeiter für einen bestimmten Zeitraum gebucht ist. Wie viel Lohn er erwartungsgemäß dafür bekommt, oder ähnliche Informationen, muss man entweder aus der Personalliste und damit aus einem weiteren Dokument entnehmen, oder im Falle der Entlohnung für den Abend sogar selbst ausrechnen.

Am Ende des Abends müssen alle Arbeitszeiten der einzelnen Mitarbeiter in das DATEV System eingegeben werden. Dies birgt zum Einen natürlich wieder eine zu verhindernde Fehlerquelle, da die Werte manuell, dies bedeutet von Hand in das andere System übertragen werden. Dabei entstehen schnell Tippfehler. Außerdem stellt das natürlich eine doppelte Eingabe von Daten dar. Erst werden die Arbeitszeiten der Mitarbeiter in den Schichtplan eingetragen und im Anschluss am Ende des Abends dann noch manuell ins Abrechnungssystem übertragen.

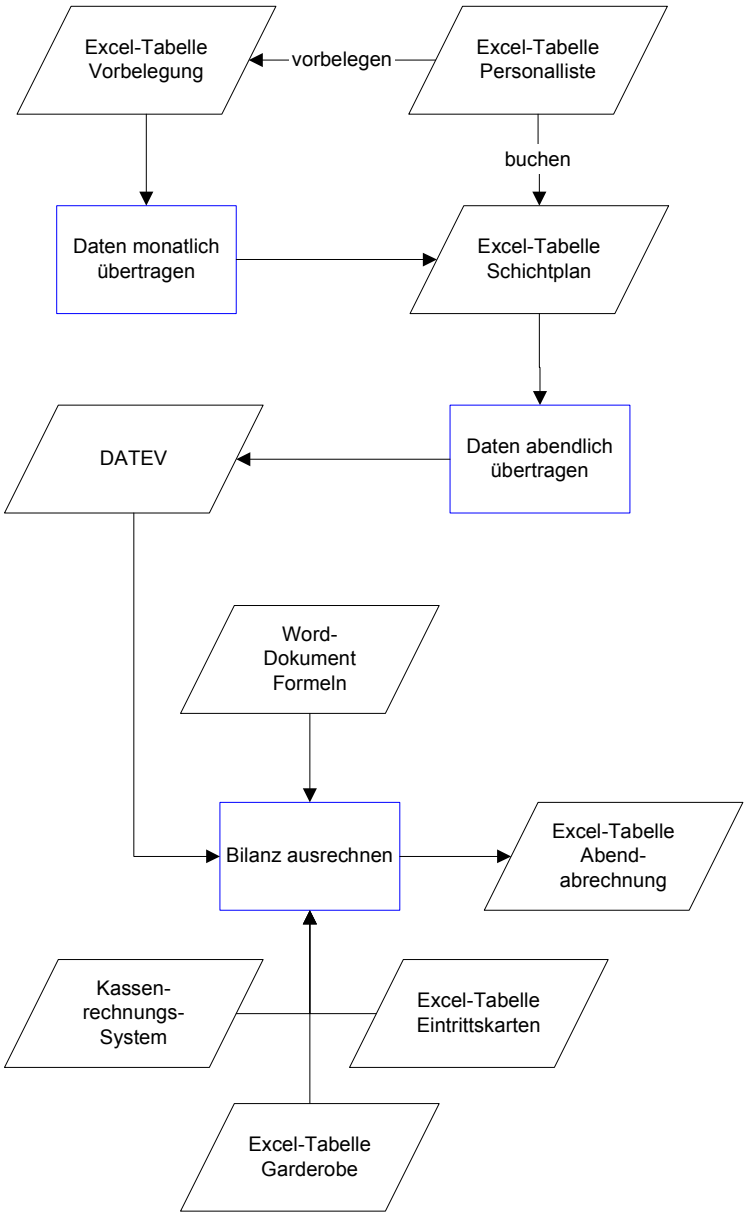


Abbildung 2-9 Ist-Zustand Abendabrechnung

Wünschenswert wäre hier, dass die Daten nur einmal in das Abrechnungssystem gebucht werden und dann nach geleisteter Arbeit lediglich freigegeben werden müssen.

Alle Daten der Einnahmen, sowie die im DATEV-System verzeichneten Ausgaben für Mitarbeiterlöhne, werden am Ende des Abends mit Hilfe der Formeln, welche in einem Word-Dokument verzeichnet sind zur Abendbilanz verrechnet.

All diese Umstände führen zu möglichen Fehlerquellen bei manuellen Übertragungen von Werten, beziehungsweise dem manuellen Ausrechnen der Bilanz. Nach dem bisherigen Schema ist es quasi unmöglich, den Ablauf transparent nach zu verfolgen. Ebenso ist es nicht möglich, verschiedene Faktoren miteinander zu vergleichen oder Statistiken kombinierter Faktoren zu erstellen und auszuwerten.

3 Ziele

3.1 Mitarbeitermanagement

Das Mitarbeitermanagement erfordert momentan die Einbeziehung vieler Arbeitsmittel, zusätzlich zum Aufwand, der manuellen Übertragung geleisteter Arbeitsstunden, von einem System ins jeweils nächste (siehe Abbildung 2-1, Abbildung 2-2 und Abbildung 2-3).

Es muss beispielsweise dafür gesorgt werden, dass beim Eintragen von Personal in den Schichtplan keine Übertragungsfehler aus der Mitarbeiterliste erfolgt. Dies wäre denkbar, wenn das Mitarbeiterkürzel falsch übertragen wird. So entstehen schnell Unstimmigkeiten, beziehungsweise Fehler. So etwas wird, wenn man den Mitarbeiter in einer Liste auswählt und auf Buchen klickt, schon im Ansatz ausgeschlossen.

Ein weiterer potentieller Gefahrenpunkt und ein unnötiger und sehr aufwendiger Arbeitsschritt ist der Schritt vom Schichtplan in die Abrechnung. Nach dem bisherigen System (siehe Abbildung 2-3) ist es nötig, jeden Mitarbeiter, der an dem betreffenden Abend gearbeitet hat, vom Schichtplan per Hand in die Abrechnung einzutippen. Neben dem Aspekt, dass dies natürlich sehr aufwendig ist, ist es völlig untransparent. Es lässt sich später auch nur sehr schwer nachvollziehen, wenn beim manuellen Übertragen irgendwo ein Fehler passiert ist und wer dafür verantwortlich ist.

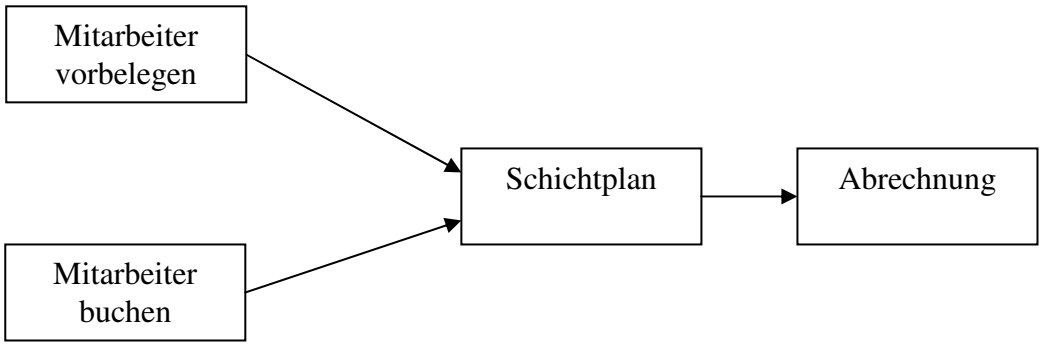


Abbildung 3-1 einheitliches Mitarbeitermanagement

3.2 Kassensystem

Wichig ist zunächst einmal, dass die fragmentierte Struktur des bisherigen Kassensystems aufgelöst wird. Das bedeutet, dass dafür gesorgt werden muss, dass es ein einheitliches System gibt, was alle Einnahmetransaktionen verwaltet, und womit man die mehrfache Haltung von Einnahmen (siehe Abbildung 2-4, Abbildung 2-5 und Abbildung 2-6) verhindert. Wie in Abbildung 3-2 werden dort also dann Artikel, also im Grunde die verkauften Getränke, die Garderobenmarken, sowie die verkauften Eintrittskarten verbucht und verwaltet.

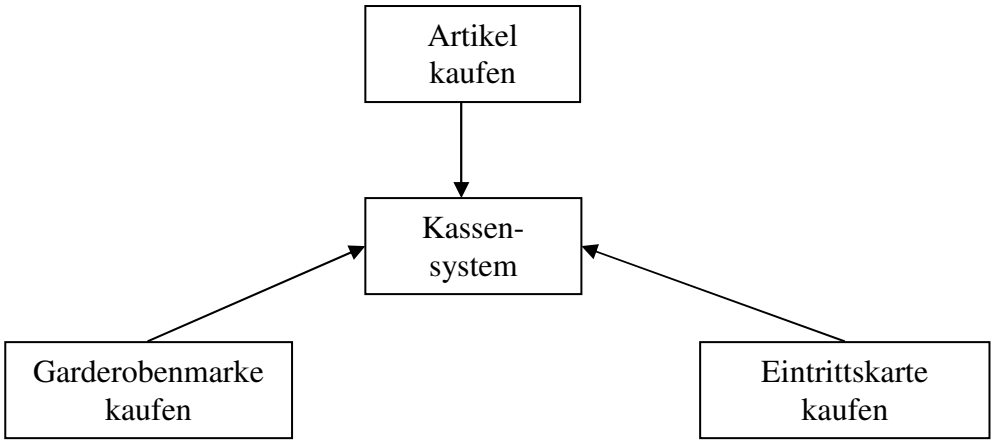


Abbildung 3-2 einheitliches Kassensystem

Hierdurch soll es später nicht mehr nötig sein, alle Einnahmen zusammen zu suchen, und diese erst noch verrechnen zu müssen, um die Bilanz erzeugen zu können.

3.3 Deejaying

Bisher ist es lediglich möglich, wenn man einen Titel, beziehungsweise einen Interpreten sucht, dies per ausgedruckter Excel Liste zu machen. Hierfür ist es nötig (siehe Abbildung 2-7), zwei Listen auf dem aktuellen Stand zu halten und wöchentlich auszudrucken. Um dieses ausdrucken zu verhindern und beispielsweise auch nach anderen Kriterien sortieren zu können, sollte die Liste inklusive Suchfunktion durch die Applikation zur Verfügung stehen.

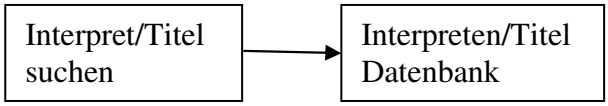


Abbildung 3-3 Interpret/Titel suchen

3.4 Abendbilanz

Bisherige Praxis war, dass am Ende des Abends alle Einnahmen, sowie alle Ausgaben aus den unterschiedlichen Systemen in ein Word Dokument eingetragen wurden (siehe Abbildung 2-8). In diesem Word Dokument wurde dann in Schritten mit den Formeln, die dort verzeichnet sind, die Abendbilanz berechnet. Dies ist natürlich mit einem gewissen Arbeitsaufwand und der Möglichkeit des Vertippens verbunden.

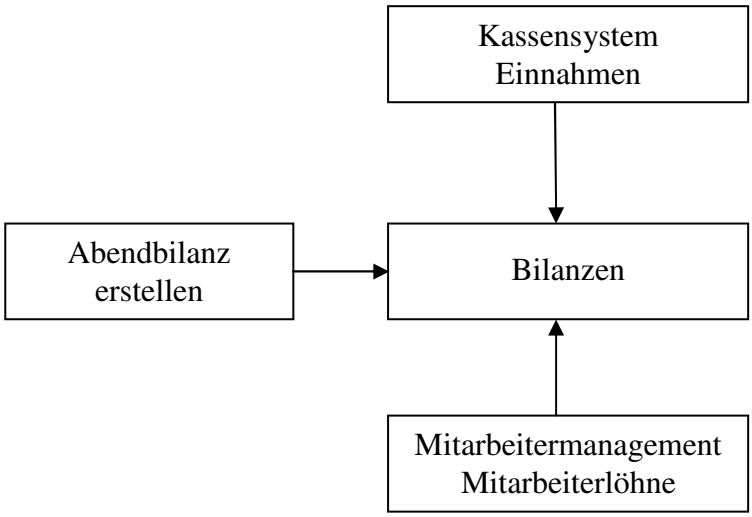


Abbildung 3-4 vereinfachte Abendbilanz

Neben der Verhinderung dieser Übertragungsarbeit ist es natürlich wünschenswert, dass all dies automatisch und ohne das manuelle Rechnen abläuft. Das System sollte also optimalerweise so wie in Abbildung 3-4 dargestellt aufgebaut sein.

3.5 Zusammenfassung

Durch die einzelnen Optimierungsschritte werden die Daten nur noch an zentralen Stellen gehalten. Das heißt, es gibt keine doppelte Haltung von ein und denselben Daten mehr, wodurch Inkonsistenzen unmöglich gemacht werden.

Wenn man sich das Modell der Abendabrechnung in Abbildung 3-5 betrachtet, fällt gleich auf, dass der Ablauf viel transparenter geworden ist. Sämtliche Stellen, an denen Werte manuell ausgerechnet oder übertragen wurden, und somit potentielle Fehlerquellen darstellten, wurden entfernt.

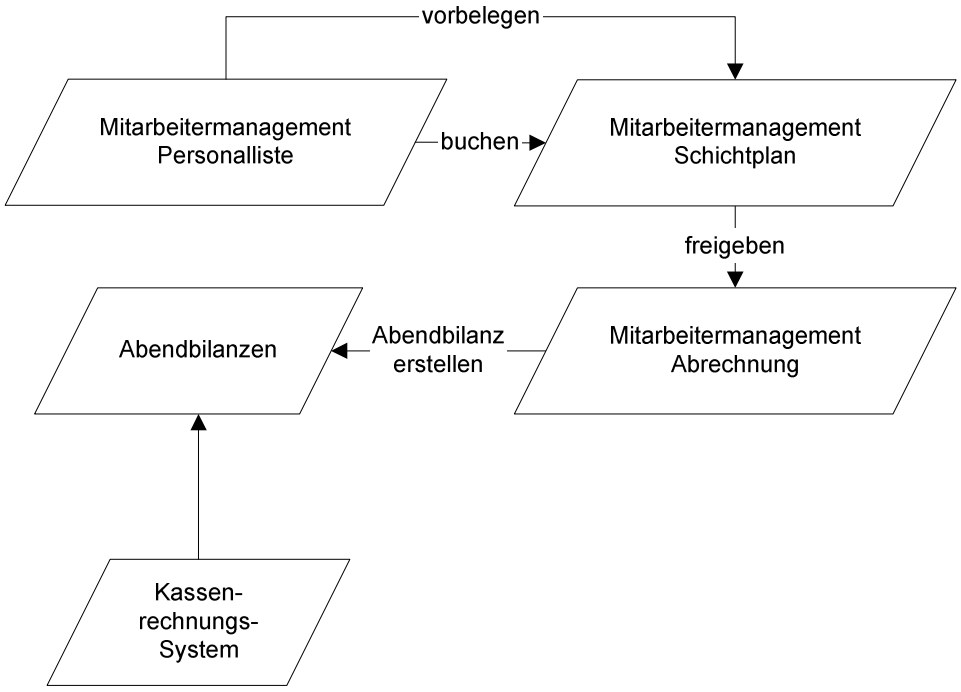


Abbildung 3-5 vereinfachte Abendabrechnung

So werden beispielsweise keine Informationen mehr vom Schichtplan in die Abrechnung händisch übertragen und die entsprechenden Arbeitslöhne dafür berechnet.

Ebenso wurde die Erstellung der Abendbilanzen beträchtlich vereinfacht. Nun werden die Abendbilanzen automatisch generiert, und es ist nicht mehr nötig, dies mit Hilfe des Formelblattes und des Taschenrechners manuell durchzuführen.

4 Pflichtenheft

4.1 Zielbestimmung

Ein Programm zur Verwaltung eines Club/Disco-Betriebes soll erstellt werden.
Es soll dazu dienen, die Aufgaben der Abrechnung (Einnahmen), des Personal-Managements und der Deejays zu erleichtern.

Musskriterien

Jedem Benutzer sollen nur die Funktionen zugänglich sein, die er zur Erfüllung seiner Aufgaben benötigt. Andere Funktionen sollen ihm verborgen sein und auch nicht indirekt zugänglich sein. Dies soll unter Nutzung einer Nutzer-Authentifizierung und –Rechteverwaltung bzw. der Trennung in eigenständige Applikationen für jeden Einsatzbereich geschehen. Die einzelnen Applikationen müssen allerdings auf einen gemeinsamen Datenpool zugreifen.

Das Programm soll ein Kassensystem beinhalten, welches das Kassieren inklusive des Rückbuchens von falsch gebuchten Artikeln ermöglicht. Des Weiteren soll eine Funktion für die abendliche Abrechnung bereitgestellt werden. Die Daten sollen während des Kassierens von jeder Kasse einzeln gespeichert werden. Am Ende des Abends muss dann für jede Kasse eine Summe gebildet werden, die dann mit dem tatsächlichen kassierten Geldstand in der Kasse verglichen wird. Ferner soll es möglich sein, einzusehen, wie viel von jedem einzelnen Artikel verkauft wurde, was die Inventur erleichtern soll.

Zur Unterstützung der Personalmanagements soll ein Personalplan erstellt werden können. Hier soll es möglich sein, individuell Personal für einen Abend zu bestimmen, aber auch Personal einzuteilen, welches an einem bestimmten Abenden regelmäßig arbeitet. Diese Informationen sollen am Ende eines Abends freigegeben, und damit in das Abrechnungsmodul übertragen werden können. Nachträgliche Änderungen an den geleisteten Stunden sollen nach der Freigabe des Tages, bzw. des Monats nicht mehr möglich sein. Eine Ausnahme davon soll

nur dann möglich sein, wenn die Freigabe des entsprechenden Tages rückgängig gemacht wurde.

Es muss möglich sein mit dem Programm während und am Ende des Abends eine ungefähre Kalkulation zu machen, wie die Bilanz des Abends ist. Das heißt, es muss in dem Programm die Möglichkeit geben, neben den variablen Kosten wie Personalkosten, auch Fixkosten, wie zum Beispiel Reinigung, GEMA[3] und Strom einzugeben und mit einzuberechnen.

Eine Programmfunktion soll den Deejay an seinem Abend, bzw. davor und danach bei seiner Arbeit unterstützen. Eine Playlist der gespielten Titel muss automatisch generiert werden. Dazu muss es möglich sein, in einem Plattenarchiv nach Titeln, bzw. Interpreten zu suchen. Diese Titel sollen dann übernommen und somit in die Playlist geschrieben werden können. Des Weiteren soll es möglich sein, Liedwünsche von Gästen zu vermerken, die dann den Abend mit der Anzahl der Wünsche versehen angezeigt werden.

Es soll ein Statistikmodul geben, in dem stundenweise, abendweise und monatlich aufkommende Daten analysiert werden können. Zu den stundenweise zu analysierenden Daten gehört beispielsweise eine Ausgabe der Menge der über den Abend pro Stunde verkauften Getränke. So soll ermöglicht werden, dass man sieht, wann am Abend am meisten an den Kassen umgesetzt wurde. Zu den abendweise gesammelten Daten gehört zum Beispiel die abendliche Bilanz. Diese soll dann in einem Monatsdiagramm analysiert werden können, wie sie sich über den Monat entwickelt hat. Monatlich aufkommende Daten sind zum Beispiel die Monatsbilanz diese Werte sollen in einem Jahresbilanz-Diagramm dargestellt werden.

[3] Die GEMA ist die Gesellschaft zur Musikrechteverwaltung. Die GEMA vertritt die Interessen der Künstler und treibt die Gebühren für gespielte oder kopierte Musikstücke ein.

Wunschkriterien

Falls die CD-Archivdaten nur in einer kommaseparierten Textdatei vorliegen, wäre es ganz praktisch, diese in die Datenbank importieren zu können.

Es könnte eine Funktion geben, mit deren Hilfe jeder Deejay seine persönlichen Charts eingeben kann und aus diesen Daten automatisch eine HTML-Seite generiert wird, welche dann auf die Internetseite des Clubs gestellt werden kann. Dazu wäre es im Grunde schon ausreichend, wenn das Programm nur den HTML-Teil der Tabelle generieren kann. Dieser kann dann über das CMS in die Club-Internetseite integriert werden.

Denkbar wäre eine Rangliste der Gästeliedwünsche anhand der man ablesen kann, welches Lied von den Gästen in der jeweiligen Woche am meisten gewünscht wurde. Eventuell könnte es auch möglich sein, dass die Gäste einen Liedtitel nur aus einer vom dem Deejay vorher zusammengestellten Liste wählen. So kann der Deejay vorher schon eine Vorselektion an möglichen Musiktiteln treffen

Falls es noch im Zeitrahmen der Diplomarbeit lösbar ist, könnten die Mitarbeiter noch zusätzliche Attribute wie z.B. Telefonnummer usw. bekommen, damit eine Telefonliste generiert werden könnte.

Abgrenzungskriterien

Das Programm soll im Deejaying-Bereich nicht das Speichern oder Abspielen von Musik ermöglichen. Es soll lediglich der Verwaltung von Musikinformationen dienen.

4.2 Produkteinsatz

Anwendungsbereich

Die Applikation wird zur Verwaltung in Club/Disco-Betrieben eingesetzt.

Zielgruppe

Die Zielgruppe für die Applikation sind Thekenkräfte, Eingangskassenpersonal, Garderobenpersonal, Deejays, Abendverantwortliche und die Geschäftsführer. Es wird davon ausgegangen, dass diese mit der nötigen Fachterminologie in ihrem jeweiligen Bereich vertraut sind.

Betriebsbedingungen

Das Produkt wird in einer Umgebung eingesetzt, in der teilweise, zum Beispiel beim Kassensystem, keine Maus verfügbar ist. Hier muss dann natürlich eine Steuerung ausschließlich über Tastatur möglich sein. Ebenso sollte die Hardware sehr robust konzeptioniert, beziehungsweise zusammengestellt werden, da im Einsatzgebiet auch Flüssigkeiten verschüttet werden könnten.

4.3 Produktumgebung

Sowohl serverseitig als auch clientseitig läuft das Produkt auf einem Server bzw. Arbeitsplatzrechnern mit graphischer Benutzungsoberfläche. An den Kassensystemen ist gegebenenfalls eine Spezialtastatur zur schnelleren Eingabe der verkauften Artikel vorhanden.

Software

Server: Auf dem Server wird Tomcat 5.0 inklusive JRE 1.4.1 und MySQL vorausgesetzt.

Client: Das Programm setzt auf dem Client einen HTML 4.0-kompatiblen Browsern voraus. Des Weiteren muss eine Netzwerkverbindung zum Server bestehen.

Hardware

Es muss ein Netzwerk eingerichtet sein, über das die Clients mit dem Server verbunden sind.

Für die Kassen ist es empfehlenswert Kassentastaturen, die die Eingabe von verkauften Artikeln erleichtert, zu benutzen.

Für den Kassenclients wird ein Monitor mit einer Auflösung von mindestens 640x480 vorausgesetzt. Die übrigen Clients müssen über eine Auflösung von mindestens 1024x768 verfügen. Diese Auflösungen entsprechen bei den Kassenclients Monitoren von mindestens 11 Zoll und bei den übrigen Clients Monitoren von mindestens 17 Zoll.

Produktschnittstellen

Alle Daten, die das Programm verwendet, sollen in eine MySQL-DB gespeichert werden. Es soll möglich sein, die geleisteten Arbeitsstunden der Mitarbeiter in eine kommaseparierte Textdatei zu exportieren, damit diese Werte von einem Abrechnungsprogramm weiter verarbeitet werden können.

4.4 Produktfunktionen

- Anlegen, Löschen und Ändern von Mitarbeitern
- Anlegen, Löschen und Ändern von Vertragsdaten
- Anlegen und Löschen von Artikeln
- Freigabe von Wochen und Monaten für die Abrechnung
- Abend- und Monatsabrechnung
- Statistik
    - Produktverkäufe
    - Besucherzahlen
    - Gewinne
- Login/Logout
- Deejaying
    - Interpreten- und Titelsuche
    - Aufschreiben von Liedwünschen
    - Gespielte Titel
- Kasse
- Personalplanung

4.5 Produktdaten

- Arbeitsbereiche:
  Welche Arbeitsbereiche es gibt.
- Arbeitskonto:
  Hält fest, wer von wann bis wann gearbeitet hat. Dies sollen dann auch die Daten sein, die für die Monatsabrechnung verwendet werden sollen.
- Artikel
  Welche Artikel es zum Verkauf gibt.
- Booking
  Hält fest, wer für welchen Tag gebucht ist. Dieses Buchen ist jedoch noch nicht mit einer Gutschrift über geleistete Stunden verbunden.
- Einnahmen
  Welche Einnahmen gab es.
- Mitarbeiter
  Hier finden sich die Mitarbeiterdaten
- Setting
  Speichert Programmeinstellungen.

4.6 Produktleistungen

Das Programm soll so geschrieben sein, dass in absehbarer Zeit und in Anbetracht der vermuteten Datenmengen keine Einschränkungen in Punkto Geschwindigkeit oder Zuverlässigkeit zu erwarten sind.

4.7 Nutzeroberfläche

Die Oberfläche soll sich in Punkto Navigation und Handling an normalen Webseiten orientieren, was die Bedienung des Programms erleichtern soll. Lediglich das Statistik- und Kassenmodul soll sich je einem Applet bedienen, was die grafische Darstellung erst ermöglicht.

4.8 Qualitätszielbestimmung

Die Korrektheit der Daten sowie die logischen Abhängigkeiten müssen gewährleistet werden. Nach der Eingabe von Daten wird überprüft, ob diese

Daten in den Eingabefeldern zulässig sind. Ein Speichern inkorrekter, bzw. unzulässiger Daten soll verhindert werden.

4.9 Globale Testfälle

- Anlegen, Löschen und Ändern von Mitarbeitern und Artikeln
- Buchung von Abendpersonal
- Freigeben von Tagen und Monaten
- Statistische Auswertung von Abenden/Wochen
- Bilanzen erzeugen
- Plattformunabhängigkeit: Die Applikation sollte auf möglichst vielen Betriebssystem und Browserkombinationen getestet werden.

4.10 Entwicklungsumgebung

- Sun JDK 1.4.1
- MySql Connector/J 3.0.11
- IBM Eclipse 3 M7
- Ultra Edit 9.00a
- MySQL 4.0.18
- MySQL Control Center 0.9.4-beta
- MySQL Administrator 1.0.1a
- Tomcat 5.0

5 Grundsätzliches

5.1 Visualisierung

Ursprünglich lagen Webseiten als reine HTML-Seiten fest kodiert auf einem Webserver, von welchem aus sie beim Aufruf durch einen Client an den Client übertragen wurden (siehe Abbildung 4-5-1 statische Seiten).

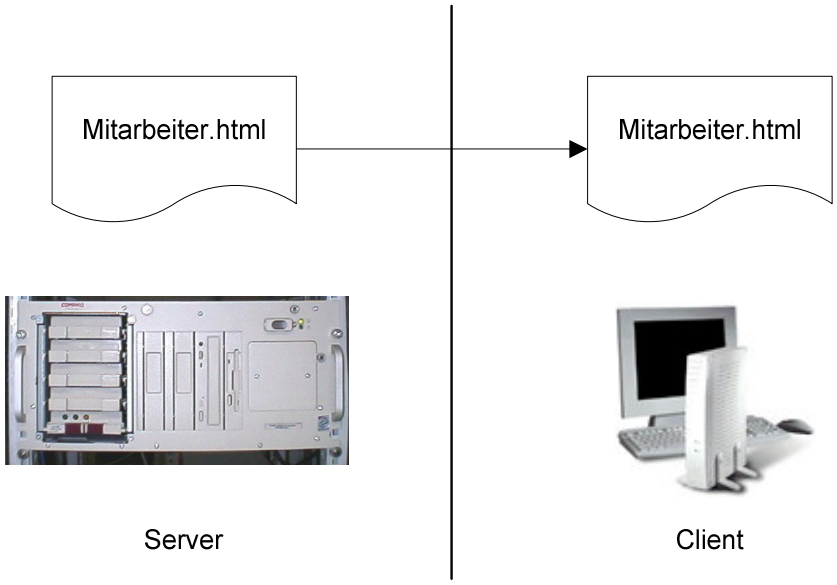


Abbildung 4-5-1 statische Seiten

Diese Art der Aufbereitung von Informationen reichte jedoch für viele Anwendungsfälle nicht aus, da dieses Konzept kaum Möglichkeiten für Interaktivität vorsah.

So mussten beispielsweise für alle vorgesehenen Ansichten einer Seite eine eigene statische Variante geschrieben werden. Dies war zum Beispiel schon der Fall, wenn es vorgesehen war, dass man ein Mitarbeiterverzeichnis nach Name oder nach Geburtsdatum sortieren kann. Das erhöhte natürlich den Pflegeaufwand der Seiten, insbesondere durch die mehrfache Haltung von Daten immens.

Neben dem Problem der mehrfachen Pflege der selben Informationen gab es noch einen weiteren Punkt, der Gefahrenpotential bot. Dies war die Tatsache, dass nicht sichergestellt war, dass wenn Informationen an einer Stelle geändert wurden, dies auch automatisch alle Stellen betraf, an denen die Information noch ausgegeben

wurde. Wenn sich beispielsweise die Adresse des Mitarbeiters geändert hat, dann muss diese Information auf allen Seiten geändert werden. Falls dies an irgendeiner Stelle übersehen wird, kommt es zu inkonsistenten Informationen.

Es mussten also Konzepte her, bei denen die Informationen an einer Stelle gelagert werden. Aus diesen Informationen und mit der gewählten Ansicht des Nutzers muss nun eine Seite generiert werden können (siehe Abbildung 4-5-2 Generierung einer dynamischen Seite).

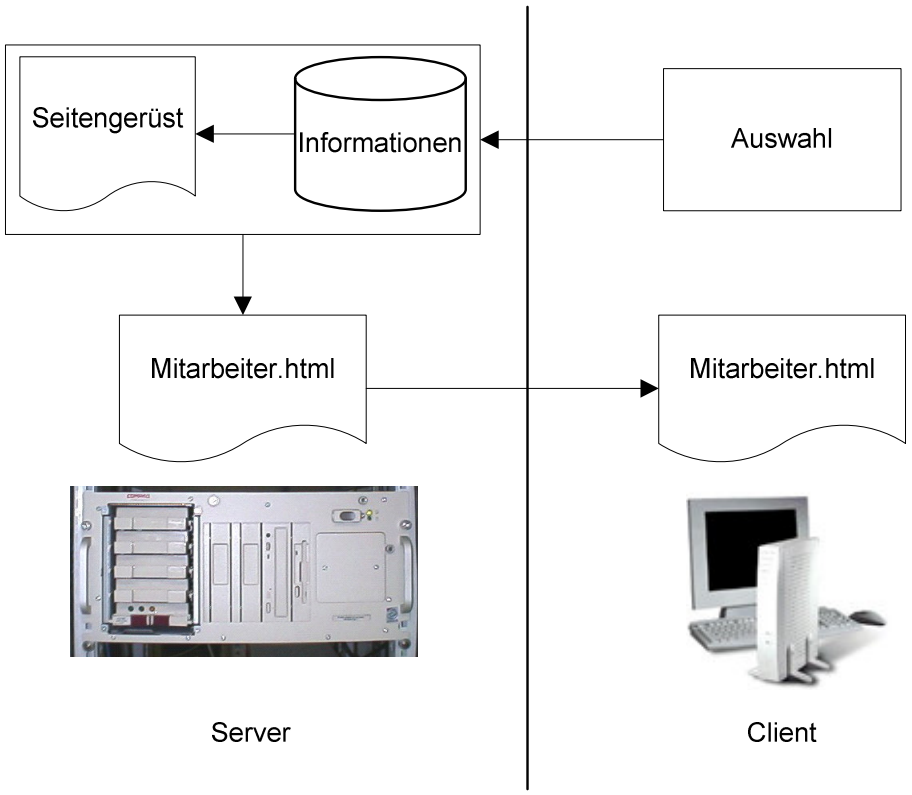


Abbildung 4-5-2 Generierung einer dynamischen Seite

Servlets

Eine erste Antwort aus der Java Ecke waren die Java Servlets. Der Begriff Servlets setzt sich zusammen aus den Anfangs- und Endbuchstaben des Begriffs "**Serv**erseitige App**lets**"

Java Servlets sind Programme welche in einem Servlet-Container über eine JVM laufen und Webseiten generieren.

```
import java.io.*;
import java.util.*
import javax.servlet.*;

public class FirstServlet extends GenericServlet{

        public void service(ServletRequest request, ServletResponse response)
                                throws ServletException, IOException {
                PrintWriter out = response.getWriter();
                out.println("<html><head><title>Hello</title></head><body>");
                out.println("Hello Word!");
                GregorianCalendar now = new GregorianCalendar();
                out.println(now.getTimeInMillis() + " since 1.1.1970");
                out.println("</body><html>");
        }
}
```

Abbildung 5-3 Ein Beispiel für ein einfaches Servlet

Wie in Abbildung 5-3 zu sehen, ist der grundsätzliche Aufbau eines Servlets recht einfach. Es wird ein Ausgabestrom erzeugt und in diesen Strom werden die einzelnen HTML-Zeilen reingeschrieben.

Was auf den ersten Blick noch recht übersichtlich erscheint, wird spätestens, wenn man die ersten dynamischen Elemente hinzufügt, unübersichtlicher. Insbesondere ist dies deshalb der Fall, da nun zwar Informationen und unterschiedliche Visualisierungen an einem Ort vereint sind, aber Information und die eine übrige Visualisierung, nicht getrennt sind.

Es wird also nach einer Technik gesucht, welche Informationen an einem Ort sammelt, die Visualisierung jedoch von den Informationen trennt.

Java Server Pages

Als Lösung für dieses Problem, wurde eine neue dynamische serverseitige Technologie entwickelt. Java Server Pages trennen die Beschaffung und Haltung, sowie die Visualisierung von Informationen.

Im Wesentlichen ist eine Java Server Page-Seite eine HTML-Seite, welche um die Möglichkeit erweitert wurde, Java Code in ihr zu verwenden und so natürlich auch Java-Klassen zu verwenden, beziehungsweise zu instantiieren. Java Server Pages werden jedoch nicht einfach an den Client ausgeliefert, sondern sie werden bevor sie ausgeliefert werden in ein Servlet umgewandelt und dann vom Servlet Container verarbeitet.

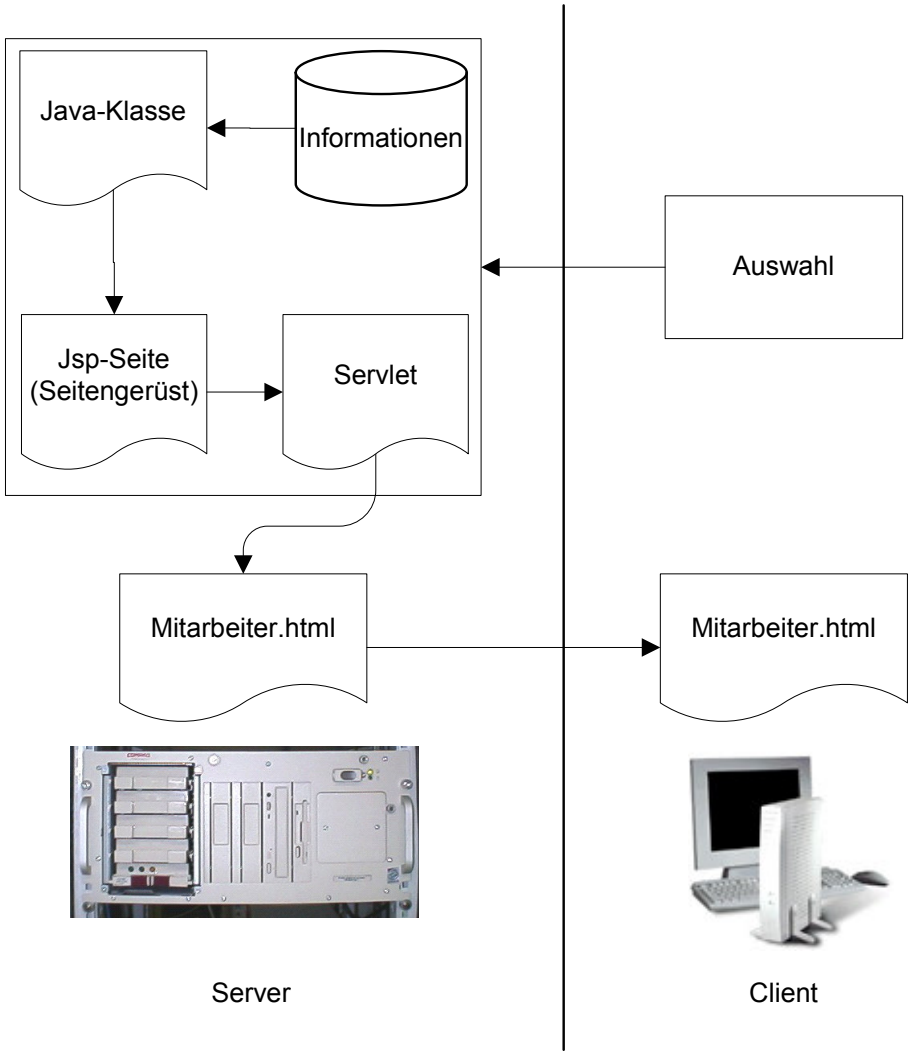


Abbildung 5-4 Ausliefern einer jsp-Seite

Beim Entwickeln einer jsp-Seite muss natürlich auch darauf geachtet werden, dass dort nicht zu viel Java Code steht, da die Übersichtlichkeit sonst wieder leidet. Ebenso verbietet es sich dort Daten beispielsweise aus einer Datenbank per jsp-Seite auszulesen und aufzubereiten. Diese Funktionen werden von einer Java Klasse verrichtet, welche die Daten dann zur Visualisierung an die jsp-Seite weiter reicht.

```
<%@ page language="java" %>
<%@ page import="java.util.*" %>

<html>
<head><title>Hello</title></head>
<body>
        Hello Word<br/>
        <%GregorianCalendar now = new GregorianCalendar();%>
        <%=now.getTimeInMillis()%> since 1.1.1970
</body>
</html>
```

Abbildung 5-5 Beispiel eine jsp-Seite

In Abbildung 5-5 sieht man, dass sich neben den normalen HTML-Anweisungen auch Java-Code in <%@ %>, <% %>, beziehungsweise <%= %> eingeschlossen befindet. In <%@ %> eingeschlossener Code ist eine Direktive, dies kann zum Beispiel eine Import Anweisung sein. In <% %> schliesst man beliebigen Java-Code ein. Mit <%= %> umschliesst man Ausdrücke. Das Ergebnis dieses Ausrducks landet an der Stelle, an der es steht, in der Ausgabeseite.

5.2 Tomcat

Um Java Server Pages zu verarbeiten und an den Client ausliefern zu können, braucht man einen Webserver mit Servlet-Container. Ein solcher Webserver ist Tomcat. Tomcat 5.0 ist die offizielle Referenzimplementierung von Suns Servlet 2.4 und JSP 2.0-Spezifikation.

Neben dem Verarbeiten und Ausliefern von jsp-Seiten bietet er auch eine Reihe webservertypischer Funktionalität, die auch ohne den optional umgebenen Apache-Server nutzbar sind. Dies sind beispielsweise das Ausliefern von normalen HTML-Seiten oder das Benutzermanagement, das heißt, dass bestimmte Seiten nur bestimmten Personen zugänglich gemacht werden.

5.2.1 Webapplikation

Die Applikation wird von Tomcat als Webapplikation verstanden, was sie im Grunde, wenn auch nicht über das Internet, auch ist. Als eine solche Webapplikation befindet sich die Applikation im Ordner /webapps unterhalb des Tomcat Root-Directory.

In diesem Ordner befindet sich gemäß der Tomcat-Spezifikation der Ordner Web-Inf. Der Ordner Web-Inf beinhaltet als wichtigstes Element in dem Unterordner classes sämtliche Klassen, welche von den Java Server Pages verwendet werden. Es ist aber auch möglich, applikationsspezifische zu nutzende Klassen als .jar-Files in den Ordner lib zu kopieren. Als weiteres wichtiges Element befindet sich in Web-Inf noch die Datei web.xml. In ihr können applikationspezifische Einstellungen bezüglich Tomcat, beziehungsweise der Webapplikation vorgenommen werden.

Neben dem Ordner Web-Inf befindet sich in der Webapplikation noch ein Ordner für die Bilder, sowie die einzelnen jsp-Seiten der Applikation (siehe Abbildung 5-6).

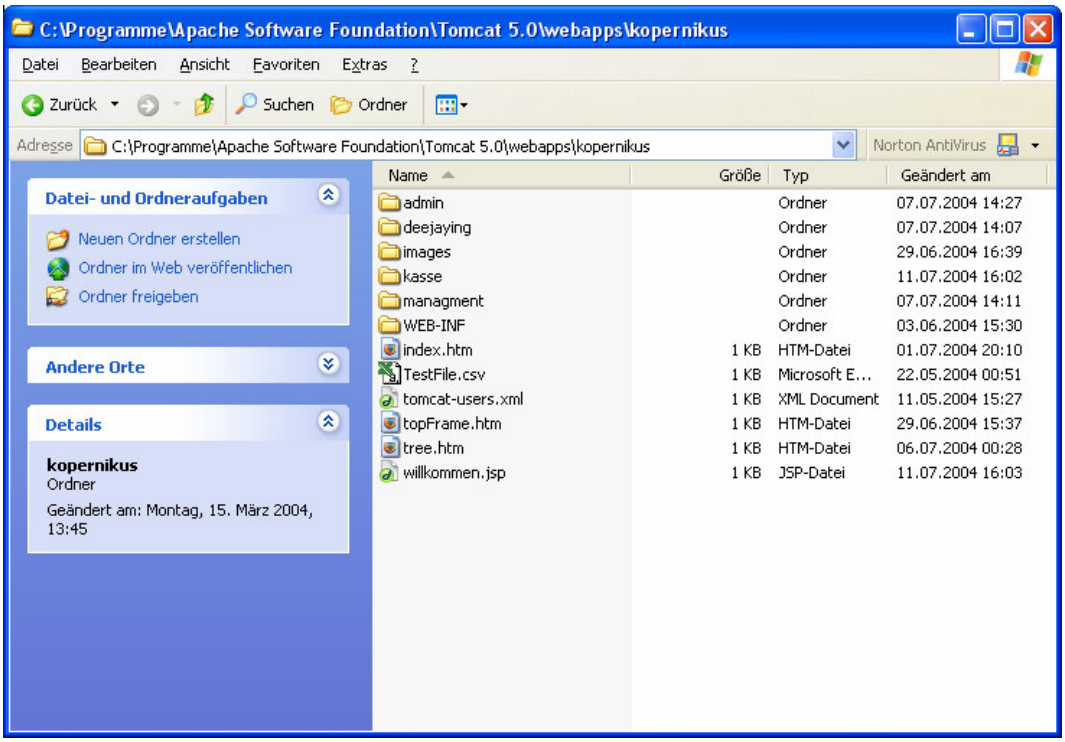


Abbildung 5-6 Ordnerstruktur Webapplikation

5.2.2 Benutzermanagement

Um nur den Nutzern Zugriff auf Komponenten zu gestatten, zu denen sie auch Zugriff haben sollen oder dürfen, ist es notwendig, ein System zur Verwaltung von Benutzerrechten zu etablieren.

Die Funktionalität, die nötig ist, um einzelne Webapplikationen, denn nichts anderes stellen die unterschiedlichen Module dar, mit einem Benutzermanagement zu versehen, liefert Tomcat schon von Haus aus.

Zunächst ist es wichtig, die Benutzer anzulegen, welche die Applikation benutzen dürfen. Hierfür kann man entweder in Tomcat 5.0 das Administrationtool[4] (siehe Abbildung 5-7) benutzen, oder aber man editiert die Datei[5] (siehe Abbildung 5-8) mit den Benutzern von Hand.

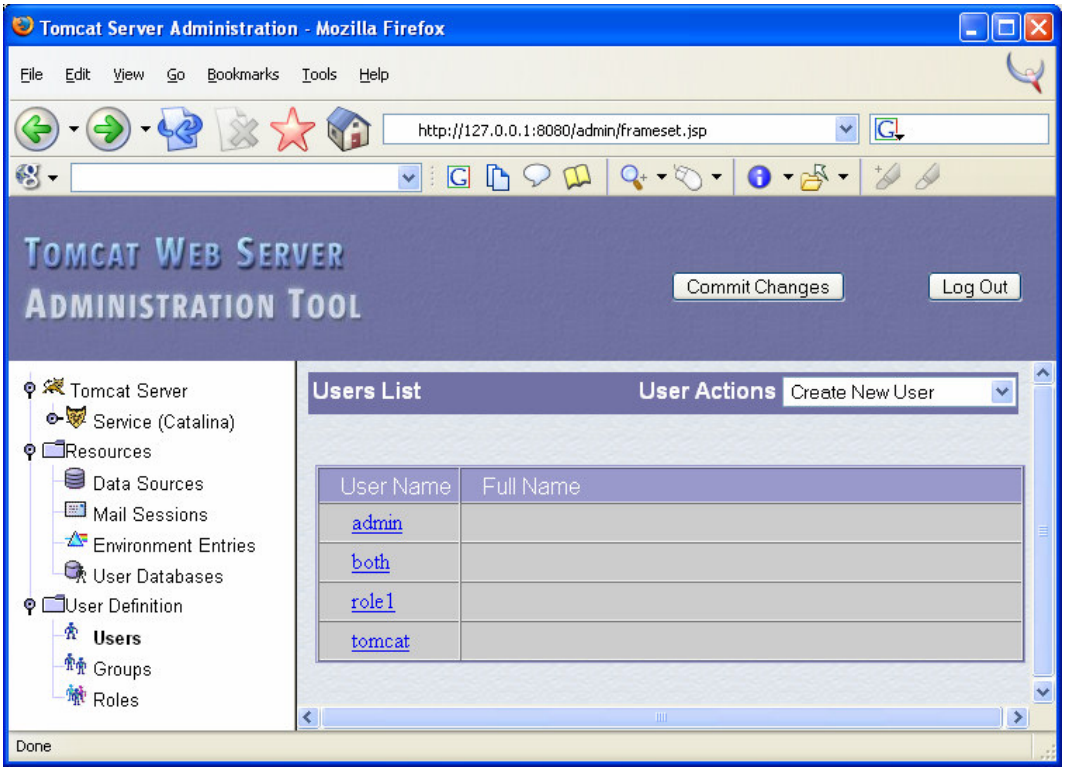


Abbildung 5-7 Tomcat 5.0 Admin-Tool

[4] erreichbar über http://localhost:8080/admin/

[5] zu finden im %TomcatHome%\conf\tomcat-users.xml

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="kassierer"/>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <role rolename="tomcat"/>
  <user username="kopernikus" password="kopernikus" roles="admin"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="theke" password="theke" roles="kassierer"/>
  <user username="both" password="tomcat" roles="kassierer,admin"/>
  <user username="bar" password="bar" roles="kassierer"/>
  <user username="admin" password="admin" roles="admin,manager"/>
</tomcat-users>
```

Abbildung 5-8 Tomcat-users.xml

Nachdem man die Benutzer angelegt hat, editiert man die Datei %Tomcat%\Webapps\[Webappname]\WEB-INF\web.xml und erweitert diese um die einzelnen Untermodule der Applikation. Hier sind dies zum Beispiel das Kassensystem-Modul oder das Admin-Modul. Die Module werden dann über die Adresse http://localhost:8080/kopernikus/kassensystem/, beziehungsweise http://localhost:8080/kopernikus/admin aufgerufen. Sie müssen selbstverständlich in der entsprechenden Hierarchie abgelegt sein.

```
<!DOCTYPE web-app
   PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
   "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
   <display-name>Kopernikus</display-name>
   <description>Kopernikus Webapplikation</description>

<!--Adminbereich-->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Kopernikus Admin</web-resource-name>
      <url-pattern>/admin/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>admin</role-name>
    </auth-constraint>
  </security-constraint>

<!--Kassensystem-->
  <security-constraint>
    <web-resource-collection>
```

```
    <web-resource-name>Kopernikus Kassensystem</web-resource-name>
    <url-pattern>/kassensystem/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
    <role-name>kassierer</role-name>
   </auth-constraint>
</security-constraint>

<!--Anmeldefenster-->
  <login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>Kopernikus</realm-name>
  </login-config>

</web-app>
```

Abbildung 5-9 Beispiel für eine server.xml

5.3 Reguläre Ausdrücke

Wenn man im Duden nachschlägt, findet man unter dem Begriff “regulär” die Entsprechung “der Regel gemäß”, “regulärer Ausdruck” bedeutet also soviel wie “ein Ausdruck, der der Regel entspricht”.

Reguläre Ausdrücke verwendet man, um festzustellen, ob in Zeichenketten bestimmte Zeichenketten existieren, oder um diese zu extrahieren[6].

Reguläre Ausdrücke bestehen aus zwei Arten von Zeichen. Zum Einen sind da die gewöhnlichen Zeichen, und zum Anderen gibt es da die Sonderzeichen. Ein gewöhnliches Zeichen passt auf eine Zeichenkette mit eben diesen gewöhnlichen Zeichen in der vorgegebenen Reihenfolge. Die Zeichenkette “ein” passt also genau auf die entsprechende Zeichenkette in “Da steht ein Pferd”, aber genauso auf das “ein” in “Das ist einmalig”.

Die Zeichen, die die regulären Ausdrücke eigentlich so mächtig und sinnvoll machen, sind die Sonderzeichen. So kann man zum Beispiel durch ein Dach am Anfang des Suchmusters bestimmen, dass die zu suchende Zeichenkette am Anfang der Zeile stehen muss. So passt “^ein” auf “ein braunes Pferd”, jedoch

[6] einen Auszug machen

nicht auf "Das ist ein braunes Pferd". Oder man kann beispielsweise überprüfen, ob in einem String großgeschriebene Buchstaben vorkommen oder nicht. Dies überprüft man dann mit einem "`A-Z`". So lässt sich auch beim Überprüfen von Formularfeldern feststellen, ob ein ungültiges Zeichen vorkommt. So darf eine Telefonnummer nur Zahlen enthalten. Man kann dies einfach durch das Suchmuster "0-9" überprüfen.

Im Anhang findet sich eine Tabelle, der man weitere Sondezeichen entnehmen kann. Weiterführende Literatur hierzu findet man zum Beispiel im CGI/Perl-Kapitel in [SelfHtml].

5.4 Mensch-Computer-Interaktion

5.4.1 Ziele

Laut H. Klocke in [InfTa] gibt es fünf wichtige Ziele der Mensch-Computer-Interaktion bei der Entwicklung einfach benutzbarer, sicherer und funktioneller Computersysteme:

- **Nutzen:**
  Die Systemfunktionalität sollte an die Arbeitsziele des Benutzers angepasst sein, das heißt, es sollte dem Benutzer die für seine Arbeit nötige Funktionalität bereitstellen, aber nicht mit für seine Ziele nicht benötigten Funktionen überfordern.
- **Effektivität**
  Damit das System einen Vorteil bringt, sollte es zu kürzeren Bearbeitungszeiten von Aufgaben führen als ohne Unterstützung durch das System.
- **Effizienz**
  Ein System mit guter Mensch-Computer Interaktion, sollte die Qualität der Aufgabenlösungen verbessern. Das heißt, es sollte genauer oder sauberer arbeiten. So kann beispielsweise dort, wo vorher ein handgeschriebener Zettel das Ergebnis war, jetzt eine sauber ausgedruckte Grafik mit einer hohen Ganauigkeit das Ergebnis sein.

- **Benutzbarkeit**
  Vielfach werden dies Systeme von Nichtinformatikern benutzt, deshalb ist eine der Grundvoraussetzungen für eine gute Interaktion die einfache Erlern- und Bedienbarkeit der Systemfunktionen. So sollte auch jemand der mit dem System noch nicht vertraut ist, mit diesem nach kürzester Zeit umgehen können.
- **Sicherheit**
  Für das System sollte es eine hohe Fehler- und Ausfallsicherheit der Software geben und gleichzeitig der Schutz privater und vertraulicher Daten beachtet werden.

5.4.2 Eingabe von Informationen

Smith und Mosier nennen in [MCIGuide] fünf allgemeine Regeln:

- **Konsistenz[7] der Dateneingabe-Transaktionen:**
  Ähnliche Eingabereaktionen unter allen Bedingungen.
- **Minimale Eingabereaktionen des Benutzers:**
  Der Benutzer soll nur die Informationen eingeben, die das System nicht von anderer Stelle beziehen kann. Keine redundante[8] Eingaben; besser die Informationen kopieren. Abwägen von Auswahlmenüs und freier Texteingabe.
- **Minimale Gedächtnisbelastung:**
  Das System unterstützt den Benutzer bei der Eingabe durch Erinnerungshilfen.
- **Kompatibilität von Informationseingabe und –darstellung:**
  Zum Beispiel gleiches Datumformat bei der Eingabe und Darstellung. Dies wird sichergestellt durch die in Kapitel 15.2.1 erklärte Klasse CalendarDate.

---

[7] Vollständigkeit, Richtigkeit und Widerspruchsfreiheit der Informationen, beispielsweise in der Datenbank. In diesem Falle ist hiermit jedoch gemeint, dass die Transaktionen immer die gleichen oder sehr ähnlichen Reaktionen der Applikation hervorrufen, damit der Benutzer sich nicht immer neu orientieren muss.

[8] überflüssige, doppelte Eingaben

- **Flexibilität bei der Informationseingabe:**

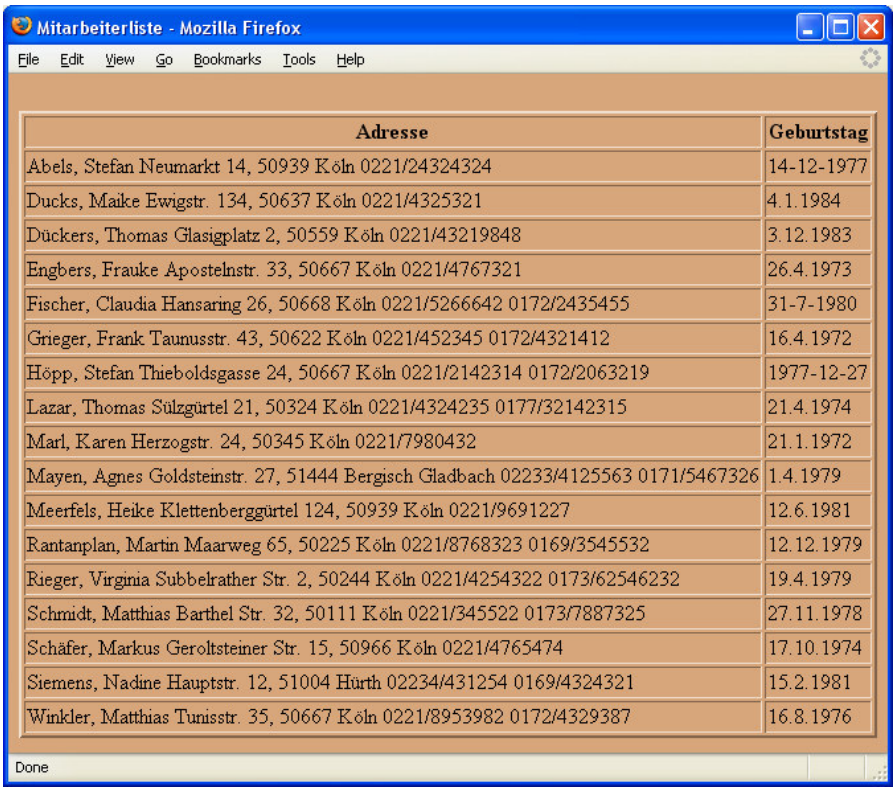
  Der Benutzer kann verschiedene Eingabeformen wie Wizards, Menüs, Funktionstasten, Befehle, Makros usw. wählen. Diese Regel ist für die Applikation nur eingeschränkt relevant, da sie eine flache Funktionshierarchie hat und alle anwendbaren Funktionen immer direkt erreichbar sind.

5.4.3 Darstellung von Informationen

H. Klocke beschreibt in [InfTa] zwei Grundregeln, welche bei der Informationsdarstellung beachtet werden müssen.

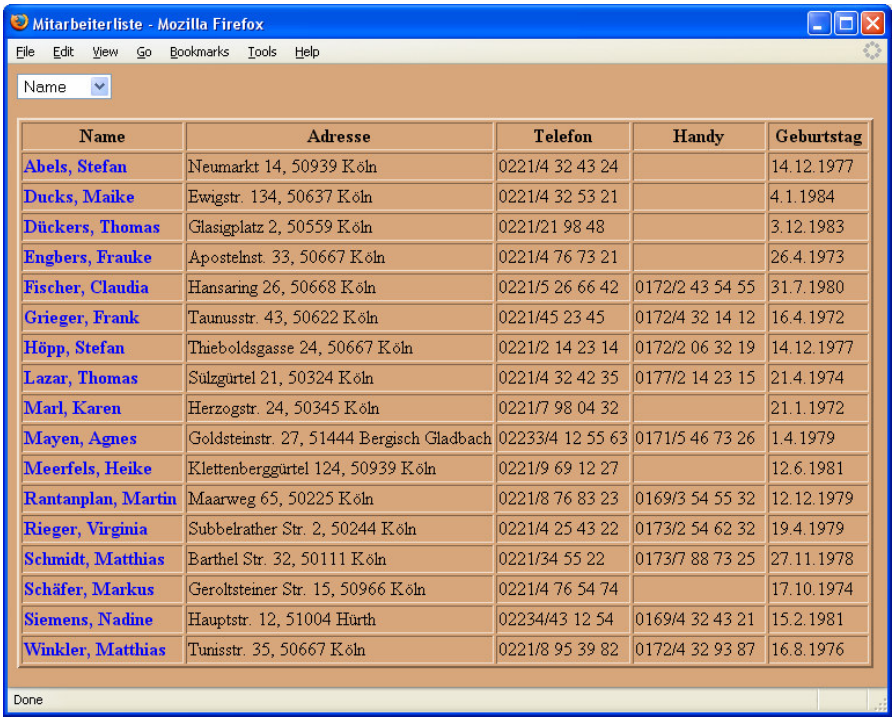
- **Organisation von Darstellungen:**
  - Konsistente und struktierte Darstellungen (Farben, Fonts, Abkürzungen usw.)
  - Minimale Gedächtnisbelastung
  - Kompatibilität von Eingabe und Darstellung
  - Flexibilität von Informationsdarstellungen

  Beispiel: Darstellung einer Kontakt- und Geburtstagsliste



Adresse	Geburtstag
Abels, Stefan Neumarkt 14, 50939 Köln 0221/24324324	14-12-1977
Ducks, Maike Ewigstr. 134, 50637 Köln 0221/4325321	4.1.1984
Dückers, Thomas Glasigplatz 2, 50559 Köln 0221/43219848	3.12.1983
Engbers, Frauke Apostelnstr. 33, 50667 Köln 0221/4767321	26.4.1973
Fischer, Claudia Hansaring 26, 50668 Köln 0221/5266642 0172/2435455	31-7-1980
Grieger, Frank Taunusstr. 43, 50622 Köln 0221/452345 0172/4321412	16.4.1972
Höpp, Stefan Thieboldsgasse 24, 50667 Köln 0221/2142314 0172/2063219	1977-12-27
Lazar, Thomas Sülzgürtel 21, 50324 Köln 0221/4324235 0177/32142315	21.4.1974
Marl, Karen Herzogstr. 24, 50345 Köln 0221/7980432	21.1.1972
Mayen, Agnes Goldsteinstr. 27, 51444 Bergisch Gladbach 02233/4125563 0171/5467326	1.4.1979
Meerfels, Heike Klettenberggürtel 124, 50939 Köln 0221/9691227	12.6.1981
Rantanplan, Martin Maarweg 65, 50225 Köln 0221/8768323 0169/3545532	12.12.1979
Rieger, Virginia Subbelrather Str. 2, 50244 Köln 0221/4254322 0173/62546232	19.4.1979
Schmidt, Matthias Barthel Str. 32, 50111 Köln 0221/345522 0173/7887325	27.11.1978
Schäfer, Markus Geroltsteiner Str. 15, 50966 Köln 0221/4765474	17.10.1974
Siemens, Nadine Hauptstr. 12, 51004 Hürth 02234/431254 0169/4324321	15.2.1981
Winkler, Matthias Tunisstr. 35, 50667 Köln 0221/8953982 0172/4329387	16.8.1976

Abbildung 5-10 schlechte Darstellung der Informationen



Name	Adresse	Telefon	Handy	Geburtstag
Abels, Stefan	Neumarkt 14, 50939 Köln	0221/4 32 43 24		14.12.1977
Ducks, Maike	Ewigstr. 134, 50637 Köln	0221/4 32 53 21		4.1.1984
Dückers, Thomas	Glasigplatz 2, 50559 Köln	0221/21 98 48		3.12.1983
Engbers, Frauke	Apostelnst. 33, 50667 Köln	0221/4 76 73 21		26.4.1973
Fischer, Claudia	Hansaring 26, 50668 Köln	0221/5 26 66 42	0172/2 43 54 55	31.7.1980
Grieger, Frank	Taunusstr. 43, 50622 Köln	0221/45 23 45	0172/4 32 14 12	16.4.1972
Höpp, Stefan	Thieboldsgasse 24, 50667 Köln	0221/2 14 23 14	0172/2 06 32 19	14.12.1977
Lazar, Thomas	Sülzgürtel 21, 50324 Köln	0221/4 32 42 35	0177/2 14 23 15	21.4.1974
Marl, Karen	Herzogstr. 24, 50345 Köln	0221/7 98 04 32		21.1.1972
Mayen, Agnes	Goldsteinstr. 27, 51444 Bergisch Gladbach	02233/4 12 55 63	0171/5 46 73 26	1.4.1979
Meerfels, Heike	Klettenberggürtel 124, 50939 Köln	0221/9 69 12 27		12.6.1981
Rantanplan, Martin	Maarweg 65, 50225 Köln	0221/8 76 83 23	0169/3 54 55 32	12.12.1979
Rieger, Virginia	Subbelrather Str. 2, 50244 Köln	0221/4 25 43 22	0173/2 54 62 32	19.4.1979
Schmidt, Matthias	Barthel Str. 32, 50111 Köln	0221/34 55 22	0173/7 88 73 25	27.11.1978
Schäfer, Markus	Geroltsteiner Str. 15, 50966 Köln	0221/4 76 54 74		17.10.1974
Siemens, Nadine	Hauptstr. 12, 51004 Hürth	02234/43 12 54	0169/4 32 43 21	15.2.1981
Winkler, Matthias	Tunisstr. 35, 50667 Köln	0221/8 95 39 82	0172/4 32 93 87	16.8.1976

Abbildung 5-11 gute Darstellung der Informationen

Konsistenz:	Identische Datumsformate und keine verschiedenen Datumsformate und Trennzeichen
Struktur:	Spaltenausrichtung für Name, Adresse, Telefon- und Handynummer
Gedächtnis:	5266642 schlecht zu merken, 5 26 66 42 kann man sich leichter merken.
Kompatibilität:	Wenn man einen Mitarbeiter anklickt und seine Details angezeigt bekommt, ist diese Darstellung genauso aufgebaut wie beim Anlegen eines neuen Mitarbeiters.
Flexibilität	Auswahlfeld zum Sortieren nach auswählbaren Kriterien.

- **Aufmerksamkeit des Benutzers:**
  Durch verschiedene Techniken wie Intensität, Markierung, Größe, Farbe, Blinken und Audio kann die Aufmerksamkeit des Benutzers erregt werden. Die Techniken sollten eingesetzt werden, um Benutzer auf bestimmte Ereignisse aufmerksam zu machen. Auf eine

kontextgerechte Integration ist dabei zu achten. Zum Beispiel ist ein akustisches Signal notwendig, wenn der Benutzer nicht permanent auf den Bilschirm schaut, aber schnell auf ein Ereignis reagieren soll.

5.4.4 Navigation

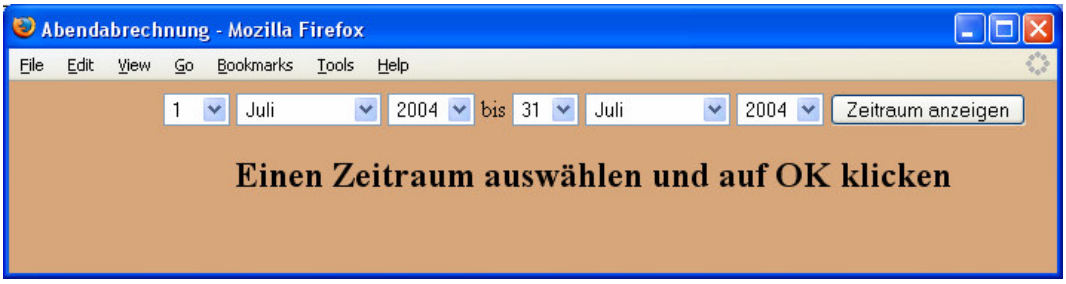


Abbildung 5-12 Schaltflächen sinngemäß beschriften

Abbildung 5-13 Baumnavigation

Schaltflächen sinngemäß beschriften

Bei Schaltflächen ist es wichtig, diese sinnvoll so zu beschriften, dass daran erkennbar ist, welche Aktionen sie auslösen. Dafür reicht es nicht, diese mit “Ok” zu beschriften, sondern es ist, beispielsweise wie in Abbildung 5-12 zu sehen, sinnvoll sie mit “Zeitraum anzeigen” zu beschreiben, wenn sie eben dies tun.

Baumnavigation

Bei der flachen Navigationshierarchie und der Vielzahl der Funktionen, welche sich jedoch gut (nach Modulen) gruppieren lassen, bietet sich eine Baumnavigation an.

Hier ist darauf zu achten, dass es stets ein Root[9]-Element gibt, welches den Sprung auf die Startseite der Applikation ermöglicht. Alternativ kann man jedoch auch, was jedoch nicht ganz so optimal ist, in jeden Dialog eine Schaltfläche zum Hauptmenü einbauen. Mindestens eine dieser beiden

[9] Root bedeutet Wurzel. Somit bezeichnet das Root-Element das Element an der Wurzel der Baumstruktur. Es kennzeichnet also das oberste Element, dem alle anderen untergeordnet sind.

Möglichkeiten ist Pflicht für eine gute Navigation.

Für die verschiedenen Arten von Elementen in der Navigation sollten unterschiedliche aussagekräftige und dem Nutzer vertraute Symbole verwendet werden. So wird beispielsweise in dieser Applikation ein Ordner-Symbol für jeden Knoten[10] verwendet.

5.5 Informationshaltung

5.5.1 Datenbank

Das Programm hat eine Menge Datensätze zu verwalten. Diese selbst in Files zu speichern und einzulesen, beziehungsweise zu schreiben und zusätzlich noch dafür zu sorgen, dass immer alles konsistent[11], nicht redundant usw. ist, ist mit einem immensen Aufwand verbunden. Diesen Aufwand kann man sich sparen, wenn man auf ein bestehendes Datenbanksystem[12] zurückgreift.

Im Rahmen dieser Diplomarbeit bietet es sich an, aus der Vielzahl der Datenbankserver MySql auszuwählen, da dieses Produkt Open Source und für private Anwendungen kostenlos ist. Für den späteren Einsatz der Applikation muss jedoch eine Datenbanklizenz gekauft werden. Diese Lizenz ist jedoch um einiges günstiger als die entsprechenden Oracle oder MS Sql Lizenz.

MySql ist auf einer Vielzahl von Plattformen verfügbar und bietet auch in Hinsicht auf Admin-Client und diverse Optimierungstools alles, was nötig und gewünscht ist.

---

[10] Die Elemente in einer Baumstruktur, welche noch Unterelemente haben, bezeichnet man als Knoten.

[11] Logik ohne Widerspruch

[12] laut L Kern-Bausch & M Jeckle in [InfTa] ein integrierter, persistenter Datenbestand einschließlich aller relevanten Informationen über die dargestellte Information (Metainformation, das heißt die Integritätsbestimmungen und Regeln), der einer Gruppe von Benutzern in nur einem Exemplar zur Verfügung steht.

5.5.2 Import/Export

Um zum Beispiel den Deejays zu ermöglichen, dass sie ihre eigenen CD-Listen im CD-Archiv der Applikation nutzen können, muss es eine einfache Möglichkeit geben, diese in die Datenbank zu importieren. Dieses Importieren darf jedoch nicht bloß über den Admin-Client der Datenbank möglich sein, da man von den Nutzern nicht erwarten kann, dieses zu verstehen und nutzen zu müssen. Abgesehen davon wäre es grob fahrlässig “normale” Benutzer Änderungen an der Datenbank direkt durchführen zu lassen.

Also muss eine einfachere Lösung, die weniger Technikverständnis vorraussetzt her. In der Applikation wurde der Weg gewählt, den Import, beziehungsweise Export, über csv-Dateien zu ermöglichen.

Csv bedeutet **c**omma **s**eparated **v**alues und beschreibt ein standartisiertes, textbasiertes Dateiformat in dem die Werte per Strich-Komma (Semikolon) separiert gespeichert werden. Diese Trennung der Werte ermöglicht ein eindeutiges Zuweisen jeden Wertes zu einer Spalte. Die Zuordnung zu Zeilen geschieht über die Zeilennummer. Dadurch lässt sich dieses Format mit allen Tabellenkalkulationen öffnen, bearbeiten und speichern.

```
Interpret;Titel;CD
Max;Can't wait until tonight;M-1
Nirvana;Smells like teen spirit;N-4
Anouk;Nobodys wife;A-2
Ace of Base;All that she wants;A-1
Rolling Stones;Angie;Best of Rolling Stones
```

Abbildung 5-14 Beispiel einer csv-Datei

6 Architektur

Das Programm ist so aufgebaut, dass es nicht nur für diesen Einsatzzweck gebraucht werden kann, sondern mit dem Grundgerüst können eine ganze Menge von Aufgabenstellungen gelöst werden. Dies wurde durch eine strikte horizontale und vertikale Aufteilung erreicht. Beispielsweise ist es für die Informationshaltungsebene egal, ob die Klassen, welche auf eine Ebene höher arbeiten, Mitarbeiter eines Clubs, oder Obst für einen Obstladen verwalten. Dadurch ist diese Ebene komplett auch für eine andere Aufgabenstellung geeignet.

6.1 Horizontal

Die Applikation ist vereinfacht dargestellt horizontal, wie in Abbildung 6-1 dargestellt, in drei Ebenen aufgeteilt.

Das Programm wird naturgemäß von unten nach oben immer konkreter. So ist die Oberfläche natürlich für jeden Einsatzzweck anzupassen, bzw. neu zu implementieren, wobei meist schon mit Anpassungen auf eine neue Anforderung reagiert werden können sollte.

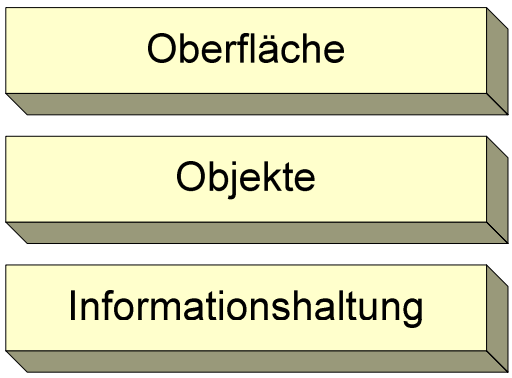


Abbildung 6-1 Horizontaler Aufbau der Applikation

Auf der obersten Ebene befindet sich die Obefläche. Diese ist mit Java Server Pages realisiert, was für die Applikation eine Menge Vorteile gegenüber der reinen Implementierung in Java bietet. Einer der Vorteile ist beispielsweise, dass die Auslieferung der Seiten, beziehungsweise der Zugriff von Clients auf die

Applikation, vom Webserver geregelt wird. Da man dies und die Benutzerverwaltung bei einer Realisierung über Java Server Pages mit Tomcat nicht selbst implementieren muss, wird eine Menge Implementierungsaufwand im Vergleich dazu, wenn man dies selber implementiert, gespart. Des weiteren ist die Darstellung von Informationen wesentlich einfacher zu realisieren, da man bei der Darstellung auf die recht einfach gehaltenen Elemente von HTML zurückgreifen kann.

Die mittlere Ebene umfasst die Objekte, welche mit dem Programm verwaltet werden. Hier sind Objekte, nicht Objekte im programmiertechnischen Sinne, sondern konkrete physische Dinge wie Gegenstände oder Personen gemeint. Wenn man das Programm nun erweitern möchte, dass es neue Objekte verwalten kann, dann muss auf dieser Ebene natürlich auch eine Erweiterung stattfinden, aber eben nicht auf der Ebene darunter. Änderungen auf dieser Ebene sind im Vergleich zur Informationshaltungsebene jedoch meist schlichterer Natur und lassen sich in Fleißarbeit realisieren.

Die dritte Ebene, die Informationshaltungsebene, ist zugleich die anspruchsvollste Ebene, da sie für beispielswesie für die Konsistenz der Daten zu sorgen hat. Diese Ebene arbeitet eng mit der Datenbank zusammen und bedient sich ihrer. Falls hier eine andere Datenbank verwendet werden soll, sind natürlich Änderungen an dieser Ebene nötig. Diese Änderung sind jedoch aufgrund dessen, dass nur rudimentäre Funktionen der Datenbank verwendet werden nicht sehr umfangreich. Damit offenbahrt sich ein weiterer Vorteil dieser Architektur. Bei Änderungen der Datenbank sind nur Anpassungen auf dieser Ebene nötig, das heißt, dass die übrigen beiden Schichten unangerührt bleiben.

6.2 Vertikal

Das Programm ist vertikal wie in Abbildung 6-2 zu sehen vertikal in sechs Schichten, vielleicht besser ausgedrückt Module, eingeteilt.

Dadurch, dass jedes dieser Module für einen eigenen isolierten Einsatzzweck zuständig ist, ist das Programm auch durch die vertikale Teilung sehr leicht

erweiterbar, beziehungsweise für einen anderen Einsatzzweck zu verwenden. Dies kann also auch vertikal, wie auch horizontal, weitgehend ohne Anpassungen an den anderen Ebenen erfolgen. Das bedeutet konkret, dass man sich die Applikation aus den Modulen zusammenbauen kann, welche man auch wirklich benötigt. Ist beispielsweise das Deejaying-Modul nicht nötig, wird es einfach weggelassen und die Applikation arbeitet auch ohne diese Probleme einwandfrei. Dadurch wird das Programm auch unabhängiger vom eigentlichen Einsatzzweck. Dem Mitarbeitermodul ist es egal, welche Art von Mitarbeitern es verwaltet und es kann sogar statt Mitarbeitern Arbeitsmittel verwaltet, der Vorgang ist im Wesentlichen derselbe.

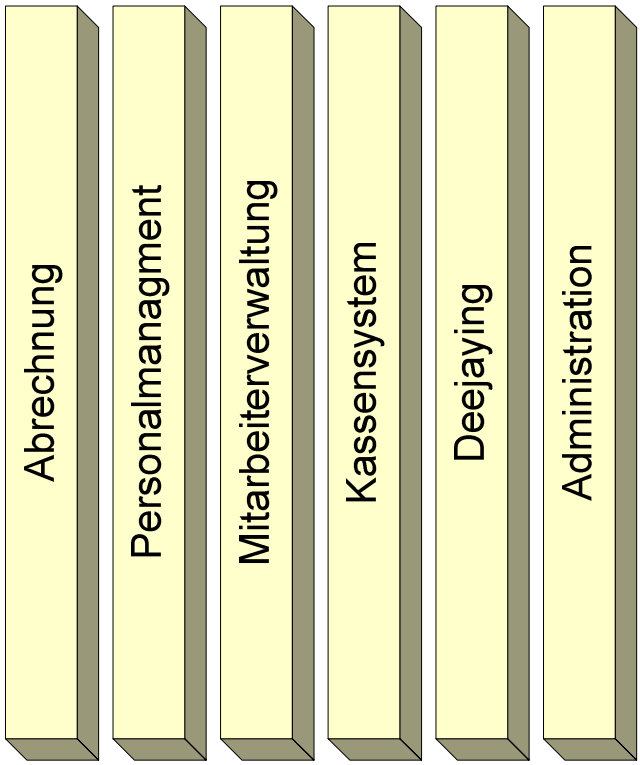


Abbildung 6-2 Vertikaler Aufbau der Applikation

Das Kassensystem stellt alle Funktionen zur Verfügung, die ein Kassensystem braucht. Wenn man mit diesem Kassensystem zum Beispiel andere Dinge verbuchen oder verwalten will, dann muss man hier nur die Objekt-Ebene anpassen. Dies muss man dann aber auch nur in der betroffenen vertikalen Ebene, sprich dem Kassensystem, tun. Man kann dieses Kassensystem durch Anpassung der oberen Schicht auch leicht umfunktionieren, beispielsweise zu einem Lagerverwaltungstool. Um das Programm für diese Aufgabenstellung zu

verwenden, müsste man lediglich die Knöpfe der Oberfläche austauschen, beziehungsweise andere Knöpfe und Schaltflächen einfügen. In diesem Falle ist es das Kassensystem, mit welchem die Thekenkräfte eines Clubs arbeiten. Es bildet ein konventionelles Kassensystem auf dem Bildschirm ab, über welches Artikel verkauft werden können.

Mit dem Personalmanagement-Modul werden alle Belange be-, beziehungsweise verarbeitet, welche mit der Verwaltung, Buchung, Entlohnung und Planung von Personal zu tun hat. Dieses Modul lässt sich auch in jedem anderen denkbaren Programm dort einsetzen, wo es um die Verwaltung von Personal geht. Das Modul ist vollkommen abstrakt vom aktuellen konkreten Einsatzzweck implementiert. In beziehungsweise mit diesem Modul lassen sich alle Arten von Angestellten anlegen, da es sich für dieses Modul ja lediglich um Bezeichnungen von Tätigkeiten handeln, die mit keiner konkreten Aktion seitens der Applikation verbunden sind.

Um seine Mitarbeiter verwalten zu können (die Mitarbeiter, nicht ihre Arbeitszeiten, Entlohnung usw.), steht das Modul Mitarbeiterverwaltung zur Verfügung. Dieses Modul verwaltet die Mitarbeiter eines Betriebes. Das heißt, es verwaltet die Adressen und die Arbeitsverträge der Mitarbeiter.

Das Abrechnungs-Modul ist das Modul, welches sich um die Abrechnung der Löhne, Steuern und Einnahmen kümmert. Mit ihm wird die Abendbilanz erstellt, oder zum Beispiel eine Auswertung der Gewinne und Umsätze angezeigt.

Mit dem Deejaying-Modul kann ein Deejay seine Aufgaben effizienter und leichter erledigen (zumindest die organisatorischen). Es hilft ihm bei der Suche von Titeln, dem Erzeugen von Playlist, oder der Wunschliste.

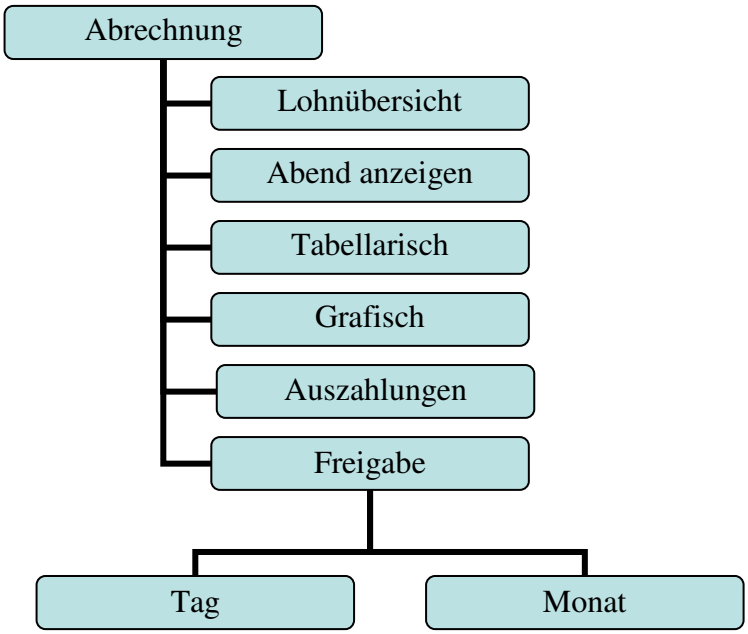
Um die administrativen Belange bearbeiten zu können, dies sind zum Beispiel das Anlegen neuer Artikel, oder Artikelgruppen, gibt es das Admin-Modul.

Jedes dieser Module ist eine eigenständige Webapplikation. Jedem der Module ist ein eigenes Kapitel gewidmet. In diesen Kapiteln befindet sich dann eine detailierte Beschreibung der Implementierung der Module.

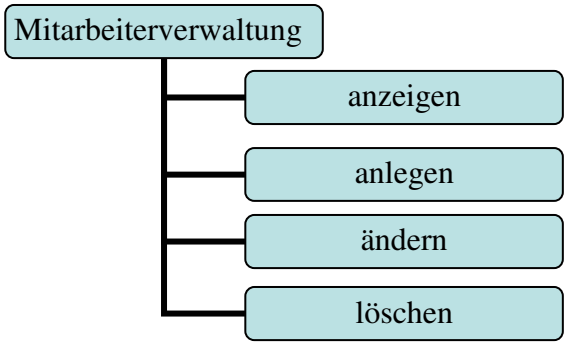
6.3 Modulfunktionsübersicht

Hier eine Übersicht über die einzelnen Funktionen, die die Module für die Applikation bereitstellen.

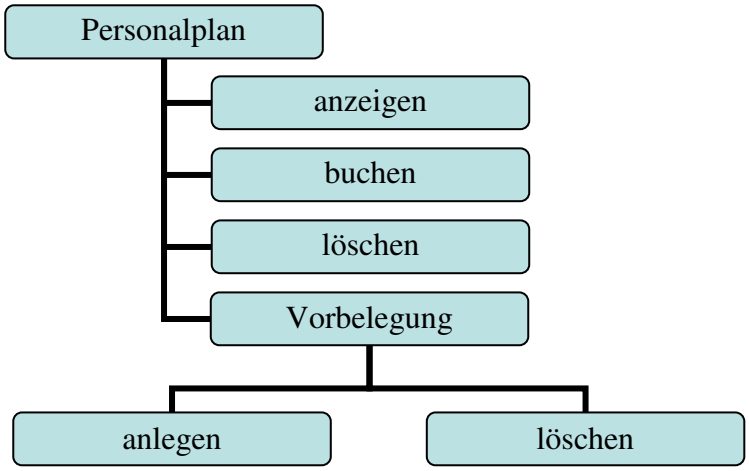
6.3.1 Modul Abrechnung



6.3.2 Modul Mitarbeiterverwaltung



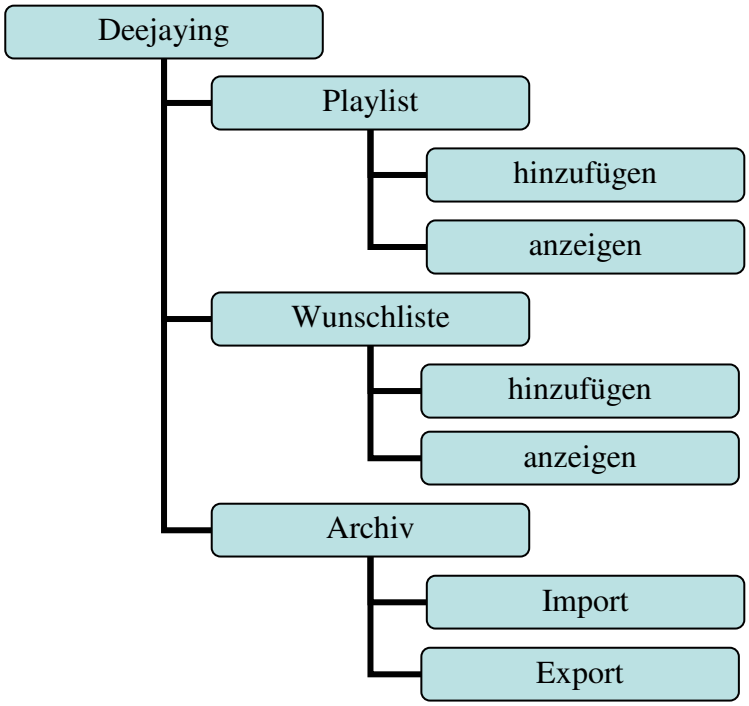
6.3.3 Modul Personalmanagement



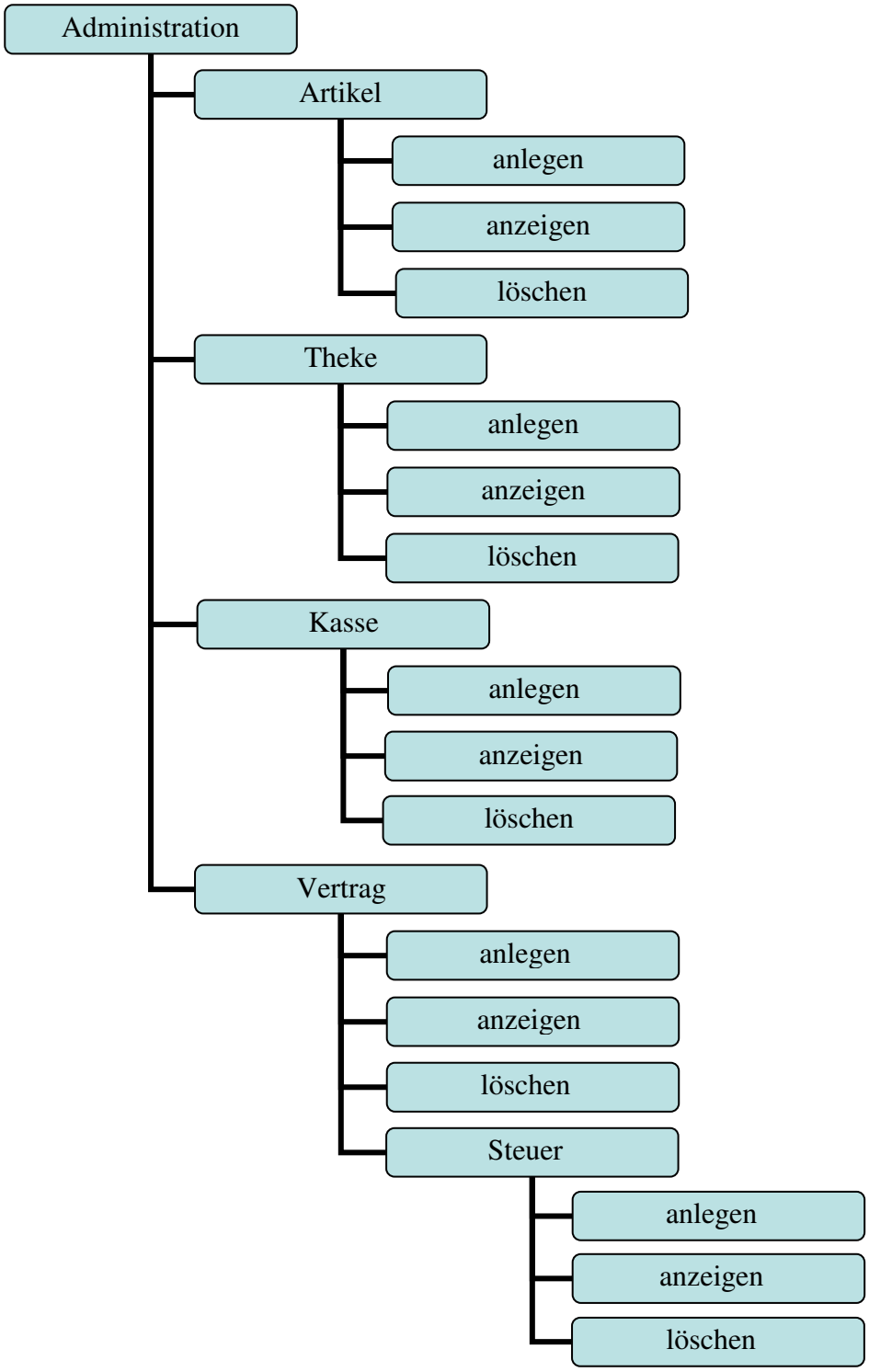
6.3.4 Modul Kassensystem

Das Kassensystem besteht nur aus zwei Seiten, welche aufeinander folgen, beziehungsweise von der zweiten gelangt man wieder auf die erste. Auf der ersten Seite meldet man sich an und auf der zweiten befindet sich dann das eigentliche Kassensystem.

6.3.5 Modul Deejaying



6.3.6 Modul Administration



7 Informationshaltung

7.1 Datenbank

7.1.1 Grundsätzliches

Um eine leichte Austauschbarkeit der Datenbank zu gewährleisten, wurde darauf verzichtet, spezielle Datumsfunktionen, oder sonstige spezielle Funktionen der Datenbank zu nutzen. In der Datenbank werden ausschließlich Strings gespeichert. Die enzige Ausnahme von dieser Praxis bilden hier die in manchen Tabellen vorkommenden Ids, bei denen es sich um selbstinkrementierende[13] Integerzahlen handelt. Bei diesen Integer-IDs ist definitiv davon auszugehen, dass sie jede für die Applikation denkbare, sei sie auch noch so beschränkt in ihrem Funktionsumfang, Datenbank, bereitstellt. Es ist auch nicht davon auszugehen, dass Datenbanken diese Datentypen unterschiedlich handhaben.

Diese Vorgehensweise hat noch weitere Vorteile. Eine davon wäre zum Beispiel, dass die Schnittstellen der Klasse SqlDataHandler sehr einfach zu halten ist, wodurch die Klasse sehr leicht anwendbar ist. Wenn man für alle Datentypen und Konbinationen die passenden get und set - Methoden schreiben würde, welche ganze Zeilen auf einmal schreiben kann, dann würde die Anzahl der Methoden dieser Klasse mit jedem möglichen Datentyp quasi quadratisch ansteigen. Wenn man ausschließlich auf Strings setzte, ist es möglich, Werte für eine Zeile auf einmal zu übergeben und in einem Rutsch zu schreiben.

Ein weiterer Vorteil ist, dass die Daten nicht zum Schreiben umgewandelt werden müssen. In fast allen Fällen liegen die Daten zum Schreiben ohnehin als String vor. Dies ist so häufig der Fall, da die eingegebenen Daten meist über mindestens eine jsp-Seite weitergereicht werden. Dieses weiterreichen erfolgt per String durch den http-Request. Wenn man also in der Datenbank nur Strings verwendet, entfällt der Schritt, diese Daten wieder in andere Datentypen zerlegen zu müssen.

[13] bei jedem Anlegen eines Datensatzes erhöht die Datenbank automatisch die nächst zu vergebene ID um eins und vergibt dieser dem neuen Datensatz.

Dies ist auch umgekehrt beim lesenden Zugriff der Fall. Die Daten werden immer, bis auf das Applet, das hier die einzige Ausnahme bildet, in jsp-Seiten angezeigt. Dafür werden die Datentypen spätestens von der Servlet Engine in Strings umgewandelt. Um zu gewährleisten, dass trotzem gültige und einheitliche Datumsstrings in der Datenbank landen, wurde die Klasse CalendarDate, welche später noch erläutert wird geschrieben.

7.1.2 mysql-connector

Um die Java Klasse mit der Datenbank verbinden zu können und Werte aus ihre lesen und in sie hinein schreiben zu können, braucht man ein Verbindungsstück. Diese Art von Verbindungsstück gibt es für viele mögliche Programmiersprachen und entsprechend auf der anderen Seite auch für jede Datenbank. Dieses Verbindungsstück stellt bei MySQL für Java der mysql-connector dar.

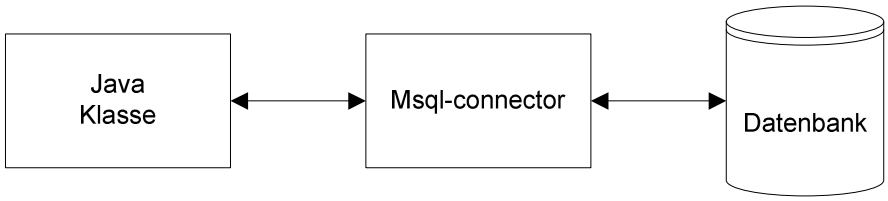


Abbildung 7-1 Zugriff auf Datenbank über msql-connector

In Abbildung 7-1 sieht man wie mit Hilfe des mysql-connector eine Verbindung zu einer MySQL Datenbank hergestellt wird. Nach dem obligatorischen Importieren der entsprechenden Klassen wird zunächst der Treiber geladen. Sobald dieser Treiber geladen ist, kann ein Objekt der Klasse Connection angelegt werden, welches die Verbindung zur Datenbank herstellt. Diesem Objekt wird zum Einen die URL, unter der die Datenbank zu erreichen ist, übergeben und zum Anderen die Benutzerdaten, das heißt Benutzer und Passwort.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class SqlDataHandler {

public SqlDataHandler(){
  try {
    Class.forName("com.mysql.jdbc.Driver").newInstance();
```

```
  } catch (Exception ex) {
    System.out.println("Treiber konnte nicht geladen werden");
  }
  try {
    String ConnectURL = "jdbc:mysql://localhost:3306/Kopernikus";
    Connection sqlConnection = DriverManager.getConnection(
                  ConnectURL,"root","");
  } catch (SQLException ex) {
      System.out.println("Keine Verbindung zur DB möglich!");
      System.out.println("SQLException: " + ex.getMessage());
      System.out.println("SQLState: " + ex.getSQLState());
      System.out.println("VendorError: " + ex.getErrorCode());
  }
}
}
```

Abbildung 7-2 Herstellen einer Verbindung mit dem mysql-connector

Sobald man eine Connection angelegt hat und somit eine Verbindung zur Datenbank steht, kann man ein Statement erzeugen über diese Connection. Dieses Statement-Objekt braucht man, um ein SQL-Statement ausführen zu können.

Es gibt zwei Arten von Methoden, die sich auf ein Statement-Objekt anwenden lassen. Da ist zum Einen eine Methode die SQL-Statements ausführt, welche eine Daten-Ergebnissmenge zurückliefern. Zum Anderen gibt es eine Methode, welche SQL-Statements ausführt, welche zu Änderungen an der Datenbank führen und nur die Anzehl der betroffenen Zeilen zurückliefern.

executeQueries

Diese Methode führt Statements aus, welche eine Ergebnismenge als ResultSet, welche im nächsten Abschnitt erklärt werden, zurückliefert. Diese Methode ist also im Wesentlichen für select-Statements relevant.

```
public ResultSet getAll(){
  try {
    Statement sqlStatement = sqlConnection.createStatement();
    ResultSet returnResultSet = sqlStatement.executeQuery(
                                  "SELECT * FROM " + tableName);
  } catch (SQLException ex) {
    System.out.println("Es konnten keine Daten gefunden werden");
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
  }
  return returnResultSet;
}
```

Abbildung 7-3 Erzeugen eines SQL-Statements

executeUpdates

Diese Methode lässt sich für Statements nutzen, welche keine Ergebnismenge als ResultSet zurückliefert. Das bedeutet, dass diese Methode auf SQL-Statements angewendet wird, welche nur die Anzahl der betroffenen Zeilen zurückliefert. Dies können zum Beispiel die Statements update oder delete sein.

```
public int delete(String paramName, String value){
  int returnInt = 0;
  try {
    String sqlString="DELETE FROM " + aTable + " WHERE " +
                     paramName + " = '" + value + "'";
    Statement sqlStatement = sqlConnection.createStatement();
    returnInt = sqlStatement.executeUpdate(sqlString);
  } catch (SQLException ex) {
    System.out.println("Es konnte nicht gelöscht werden");
    System.out.println("SQLException: " + ex.getMessage());
    System.out.println("SQLState: " + ex.getSQLState());
    System.out.println("VendorError: " + ex.getErrorCode());
  }
  return returnInt;
}
```

Abbildung 7-4 Erzeugen eines executeUpdate

ResultSet

Jede Anfrage an die Datenbank, welche Datensätze zurückliefert, liefert diese per ResultSet. Ein ResultSet ist eine Art Mischung aus einer mehrdimensionaler List und einem Iterator, es ist also eine Art List mit eingebautem Iterator.

```
ResultSet allResultSet = sqlData.getAll();
try {
  while(allResultSet.next()){
    String tmpName = allResultSet.getString("Name");
    String tmpVorname = allResultSet.getString("Vorname");
    String gebDatumString = allResultSet.getString("Gebdatum");
    CalendarDate tmpGebDatum = new CalendarDate(gebDatumString);
    ...
    Mitarbeiter tmpMitarbeiter = new Mitarbeiter(tmpName,
                                                 tmpVorname,...);
  }
} catch (SQLException e) {
  System.out.println("ungültiges ResultSet, Mitarbeiter-Klasse");
  e.printStackTrace();
}
```

Abbildung 7-5 Beispiel für die Benutzung eines ResultSet

Wenn man nun ein ResultSet iterieren möchte, macht man dies mit while(einResultSet.next()). Im Gegensatz zu einer List liefert die Methode next() hier nicht das nächste Element, sondern liefert nur einen boolean zurück, ob es

weitere Elemente gibt. Falls es weitere Elemente gibt, kann man sich die einzelnen Strings, falls es sich bei dem betreffenden Element um einen String handelt. nun mit einResultSet.getString() auslesen.

7.1.3 Anfragen parsen

Damit man übergebene Strings in SQL-Anfragen benutzen kann, muss der übergebene String vorher geparst werden. Das bedeutet, er muss um einige unerlaubte Sonderzeichen bereinigt werden. Diese Sonderzeichen müssen dann durch eine Escape-Sequenz, welche das Sonderzeichen repräsentiert, ersetzt werden. Warum dies geschehen muss, hat vielfältige Gründe, welche weiter unten explizit erläutert werden.

Um einen String parsen zu können stellt Java den Strings eine Methode zur Verfügung. Mit Hilfe dieser Methode und zur Hilfenahme eines regulären Ausdrucks (siehe Kapitel 5.3), lassen sich die ungültigen Sonderzeichen ersetzen. In Abbildung 7-6 sieht man wie die Methode replaceAll angewendet wird. Diese bekommt als ersten Parameter den regulären Ausdruck und als zweiten das übergeben, durch was er das, auf was er matcht[14], ersetzen soll.

```
public String parseString(String aStringToParse){
  aStringToParse=aStringToParse.replaceAll("\\\\","/");
  aStringToParse=aStringToParse.replaceAll("\'","\\\\'");
  return aStringToParse;
}
```

Abbildung 7-6 Parsen eines Strings

Böswilliger Angriff

Als sicherlich gewichtigster Grund, warum Strings, welche in SQL-Statements eingefügt werden, geparst werden müssen, ist der der Problematik der SQL-Injection (siehe [Injection]). Diese Problematik entsteht primär, wenn man zulässt, dass Benutzer beispielsweise auf eine jsp-Seite in ein Text-Inputfeld[15] innerhalb eines Forms[16] ihren eigenen Text eingeben (siehe Abbildung 7-7).

[14] als matchen bezeichnet man das Passen eines regulären Ausdrucks auf mindestens einen Bereich dessen, was er untersucht.

[15] Das sind auf Webseiten die Felder, in die man normalen Text eingeben kann (zum Beispiel die

Abbildung 7-7 Eingabe in ein Text-Input-Feld

Auf der Empfängerseite, welche das Form entgegen nimmt, wird der Text in einen SQL-String eingebettet (siehe Abbildung 7-10). Wenn der Nutzer in das Formularfeld etwas ungefährliches wie seinen Namen eingibt, lautet der SQLString:

```
sqlSting="DELETE FROM Kunden WHERE Name='Karl Klammer'.
```

Abbildung 7-8 ungefährliche Formulareingabe

Der potentielle Angreifer kann das freie Text-Eingabefeld aber auch dazu nutzen als Namen ‘;DROP TABLE Kunden; einzugeben. Wenn dies ohne zu parsen in den SQL-String eingefügt wird, ergibt sich daraus:

```
sqlString="DELETE FROM Kunden WHERE Name=";DROP TABLE Kunden.
```

Abbildung 7-9 gefährliche Formulareingabe

Wenn man dies nun ausführt, interpretiert SQL dies als zwei Befehle. Der erste ist ordnungsgemäß durch das zweite Hochkomma abgeschlossen, liefert nur kein

---

Eingabe eines Namens beim Neuanlegen eines Mitarbeiters).

[16] Ein Form ist ein HTML-Forular, dass zum Beispiel Input Felder oder Auswahlboxen enthält, welche an eine weitere Seite geschickt werden.

Resultat zurück, und der zweite wird durch ein Semikolon separiert eingeleitet und ausgeführt. Dem Angreifer wird also ermöglicht, beliebige SQL-Befehle auszuführen, was allerdings durch mangelndes Wissen über die Datenbankstruktur noch etwas gebremst wird. So muss er beispielsweise für die Ausführung des drop-Statements wissen, welche Tabellen die Datenbank enthält.

Durch das Parsen von Hochkommas und anderen gefährlichen Zeichen wird diese Problematik komplett entschärft, da der value-Teil[17] im SQL-String keine Hochkommas in Reinform mehr enthält, sondern nur noch die entsprechende Escape-Sequenz.

```
String value = request.getParameter("Name");
String sqlString="DELETE FROM Kunden WHERE Name='" + value + "'";
```

Abbildung 7-10 SQL-Injection Schwachstelle

Natürliches Vorkommen

Neben dem mutwilligen böswilligen Angriff gibt es noch Fälle, in denen Sonderzeichen naturgemäß vorkommen. Dies wäre zum Beispiel häufig im CD-Archiv der Fall. Viele der zumeist englischen Titel beinhalten, wie zum Beispiel “Phil Collins – You can't hurry love”, ein Hochkomma welches den SQL-String unterbrechen würde.

7.1.4 Datenbank administrieren

MySQLControlCenter

Das MySQLControlCenter wird dafür verwendet, Queries[18] an die Datenbank zu stellen, oder neue strukturelle Änderungen an der Datenbank durchzuführen. Es bietet viele Funktionen, die per SQL möglich sind als grafische Funktion, wobei es allerdings natürlich trotzdem möglich ist, dies per textuellem SQL-Statement zu tun.

[17] der Teil mit dem Wert des Attributs

[18] eine Suchanfrage an die Datenbank

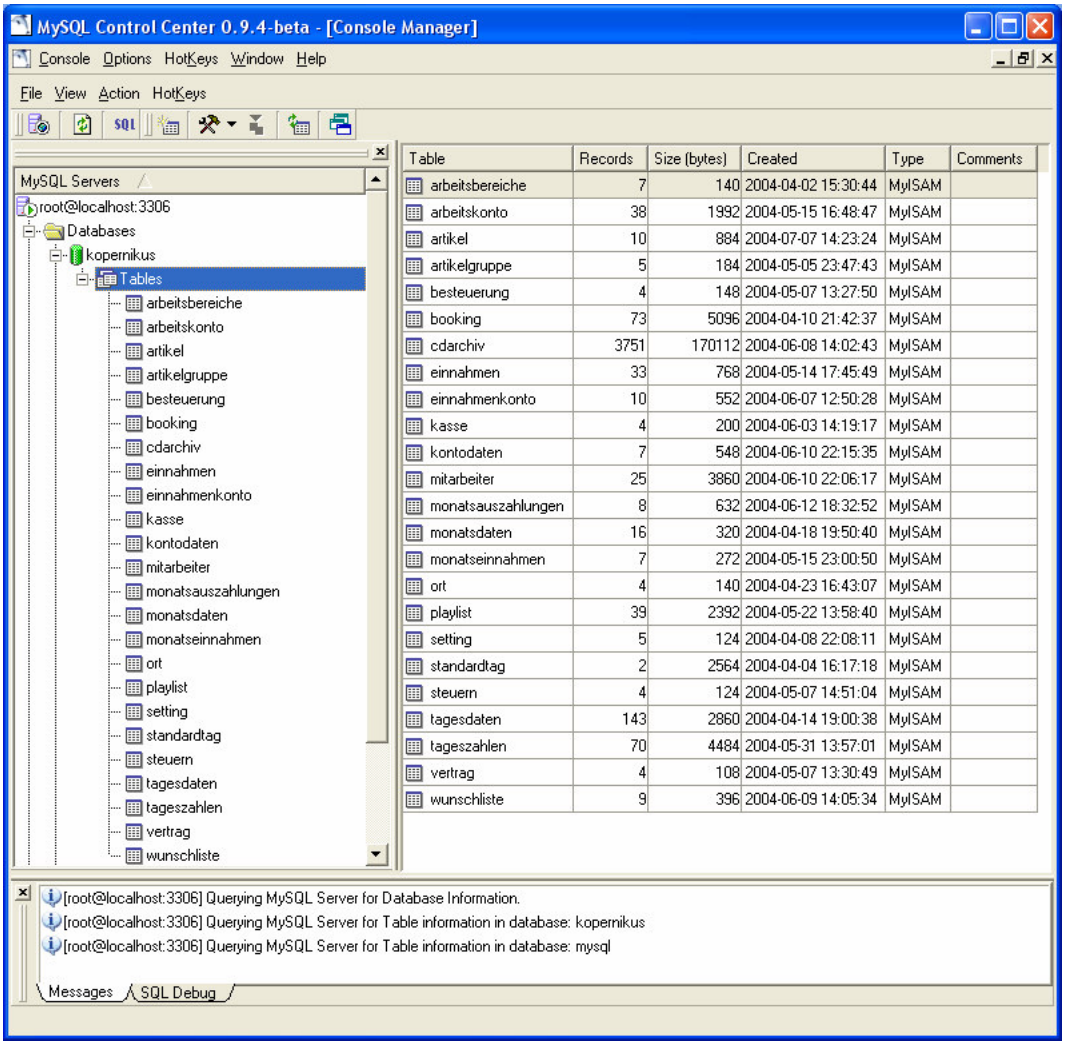


Abbildung 7-11 Das MySQLControlCenter

MySQLAdministrator

Der MySqlAdministrator stellt die administrativen Funktionalitäten bereit, die man zum Betrieb einer MySQL-Datenbank benötigt. So ist es mit dem Administrator beispielsweise möglich, eine Sicherung der aktuellen Datenbank anzulegen, beziehungsweise Informationen aus einer bestehenden Sicherung wiederherzustellen.

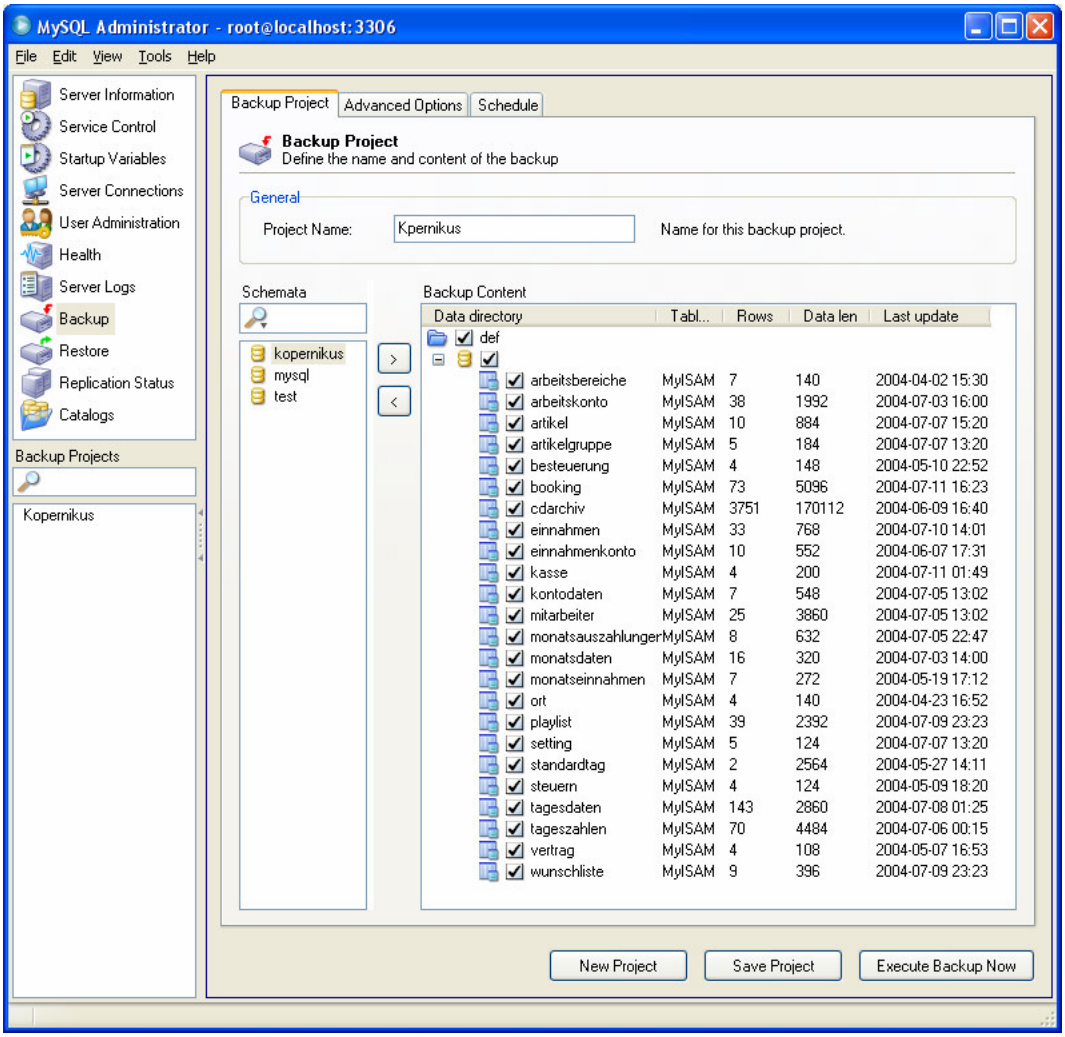


Abbildung 7-12 Der MySQLAdministrator

7.2 Files schreiben und lesen

Neben den Zugriffen auf die Datenbank ist es auch erforderlich, auf Dateien zugreifen zu können. Diese Funktionalität wird zum Beispiel für den Import und Export von Datensätzen durch das Programm benötigt. Hierfür wurde die Klasse FileDataHandler (analog zu SqlDataHandler) geschrieben. Diese Klasse kapselt alle Zugriffe auf Dateien.

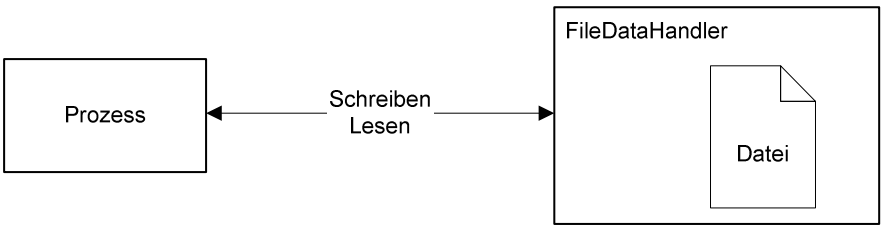


Abbildung 7-13 FileDataHandler kapselt den Dateizugriff

Schreibender Zugriff

Für den schreibenden Zugriff auf eine Datei braucht man zunächst einen FileWriter. Dieser FileWriter dient dazu, den Zeilenstrom[19] zu handhaben. Es gibt zwei Möglichkeiten einen FileWriter zu instantiieren. Die eine, bei der man nur den Dateinamen übergibt, legt eine Datei neu an, wenn sie nicht schon da ist (dann wird sie gelöscht), und die andere, der man ein zusätzliches true übergibt, hängt die neuen Zeilen an die alte Datei an.

Das eigentliche Schreiben der Zeilen erfolgt einfach, indem man die write-Methode der Klasse FileWriter anwendet und ihr die zu schreibende Zeile als String übergibt. In diesem Beispiel werden die einzelnen Zeilen an die Methode, welche die Datei schreibt per ArrayList[20] übergeben, iteriert[21], gecastet[22] und dann zeilenweise in die Datei geschrieben.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;

public class FileDataHandler {

  private String filename;

  public FileDataHandler(String filename){
    this.filename=filename;
  }

  public void store(ArrayList aZeilenList){
    Iterator zeilen = aZeilenList.iterator();
    try {
      FileWriter file = new FileWriter(filename);
      while(zeilen.hasNext()){
        String tmpZeile = (String)zeilen.next();
        tmpZeile = tmpZeile + "\n";
        file.write(tmpZeile);
      }
      file.close();
    } catch(FileNotFoundException e) {
```

[19] Als Zeilenstrom bezeichnet man die Zeilen, welche geschrieben werden. Diese werden hintereinander, das heißt als Strom in die Datei „geschoben“

[20] Eine ArrayListe ist eine Liste ähnlich einem Vector aus Java-Objekten

[21] Iterieren einer Liste bedeutet, diese Liste elementweise zu durchlaufen.

[22] Als casten bezeichnet man das implizite Umwandeln von einem Datentyp in einen anderen Datentyp

```
      e.printStackTrace();
    } catch(IOException e) {
      e.printStackTrace();
    }
  }
}
```

Abbildung 7-14 In eine Datei schreiben

Lesender Zugriff

Für den lesenden Zugriff auf eine Datei gibt es analog zum FileWriter einen FileReader. Dieser FileReader bekommt einen Dateinamen übergeben und wird seinerseits einem BufferedReader übergeben. Der BufferedReader erlaubt es alle Zeilen einzulesen und diese dann zu iterieren. Im Beispiel in Abbildung 7-15 sieht man, dass die Zeilen so lange iteriert und in die Rückgabe-ArrayList geschrieben werden bis keine Zeile mehr vorhanden ist.

```
public ArrayList getAll(){
  ArrayList linesList = new ArrayList();
  String line;
  try {
    BufferedReader bufferedreader = new BufferedReader(
                                        new FileReader(filename));
    while((line = bufferedreader.readLine())!= null){
      linesList.add(line);
    }
    bufferedreader.close();
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
    System.out.println("Fehler beim Einlesen von " + filename);
  }
  return linesList;
}
```

Abbildung 7-15 Aus einer Datei lesen

7.3 Import/Export

Der Import, beziehungsweise Export, dient in der Applikation in erster Linie dem Zweck, CD-Archiv Daten exportieren zu können. Dies ist dafür nöktig, dass Deejays ihre eigenen CD-Archive importieren, beziehungsweise das Gesamt-Archiv für sich exportieren zu können.

Die Import/Export-Methode des CD-Archivs muss darauf achten, dass die beim Lesen mit eingelesene Überschiftszeile nicht mit in die Datenbank geschrieben

wird und das anders herum beim Schreiben in die Cvs-Datei die Spalten wieder eine Überschrift bekommen. Als Separator wird wie schon in Kapiutel 5.5.2 erkläutert je ein Strichkomma (Semikolon) zum Separieren der einzelnen Zeilen verwendet.

Zum Schreiben des preparierten fertigen Satzes von Zielen verwendet die Import/Export-Methode die in Kapitle 7.2 beschriebene Klasse FileDataHandler.

Abbildung 7-16 Import/Export aus der Datenbank

8 Programmstart

Nach dem Programmstart ruft die Startseite der Applikation, auf der man die Unterapplikation auswählt, die Methode init() der Klasse Programmstart auf. Diese Methode nimmt eine Reihe von Updates, beziehungsweise Inserts auf der Datenbank vor.

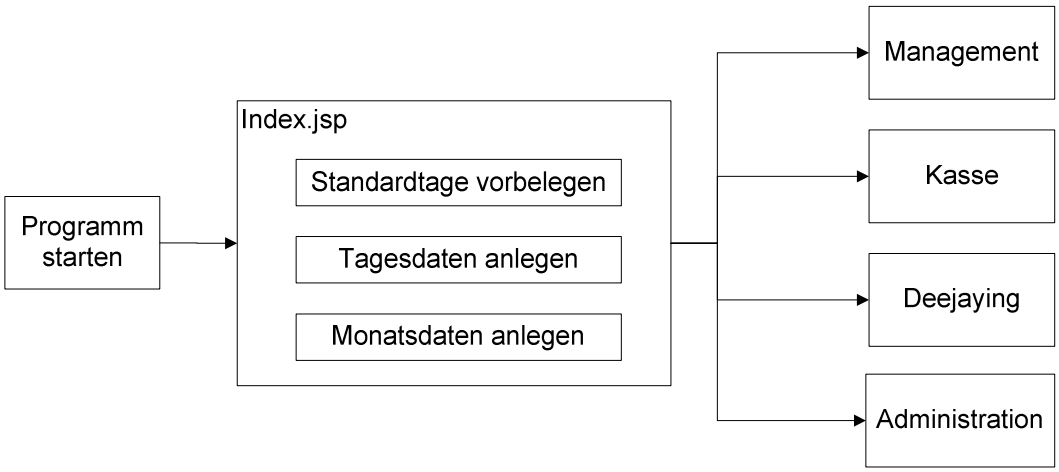


Abbildung 8-1 Programmstart

Zunächst einmal wird das Muster der aktuelle Tagesvorbelegung (vgl. Kapitel 10.4) auf die nächsten 90 Tage Booking abgebildet. Dies erfolgt natürlich nicht neu über die nächsten 90 Tage, sondern nur für die Tage, auf welche die Tagesvorbelegung noch nicht abgebildet ist. Dafür überprüft die Methode Tagesvorbelegung.preselect(), welche von init() aufgerufen wird und die Vorbelegung der Tage vornimmt, bis wann bei der letzten Vorbelegung vorbelegt wurde.

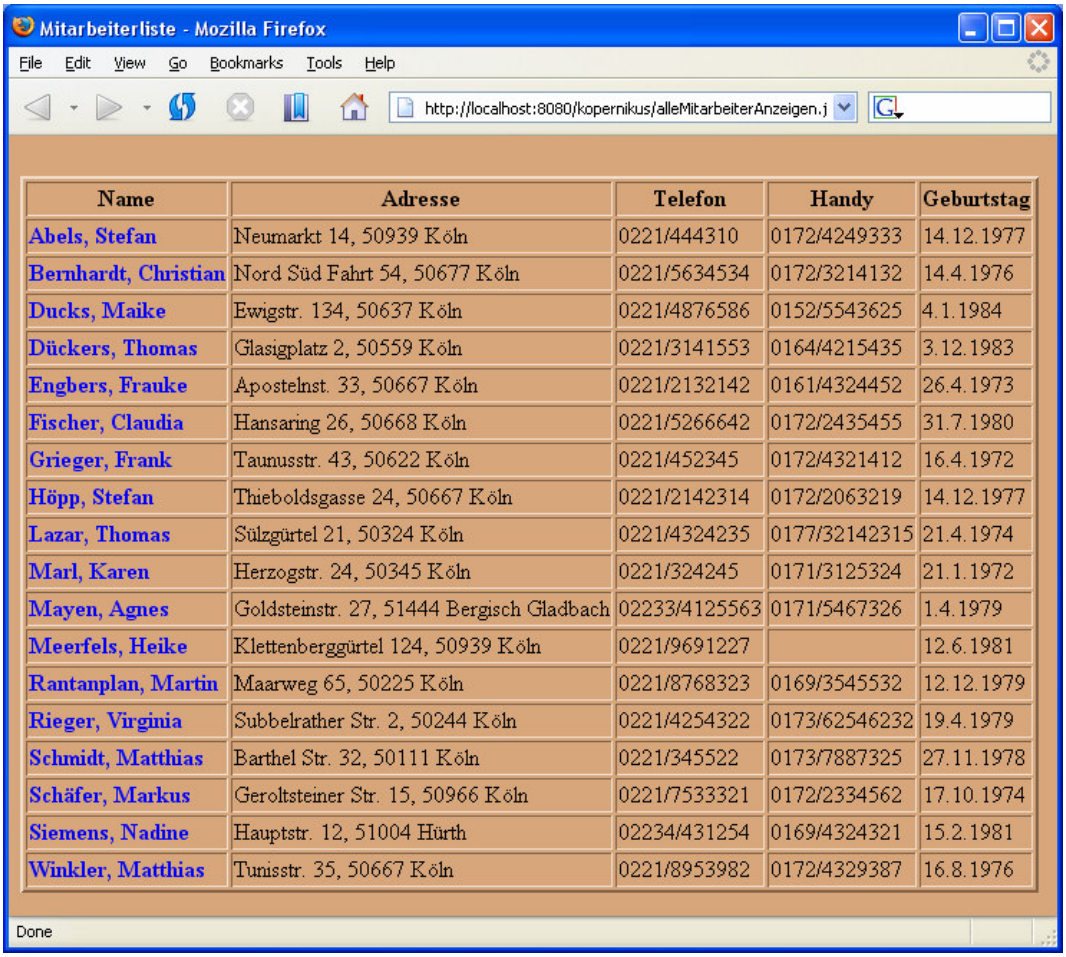
Als nächstes wird in der Datenbank durch die Methode Tagesdaten.preselect() dafür gesorgt, dass für jeden der nächsten 90 Tage für jeden Tag ein Datensatz inklusive des Wertes false für freigegeben existiert.

Entsprechend werden die nächsten zwölf Monate von der Methode Monatsdaten.preselect() wie bei den Tagen durch die Methode Tagesdaten.preselect(), vorbelegt.

9 Modul Mitarbeiterverwaltung

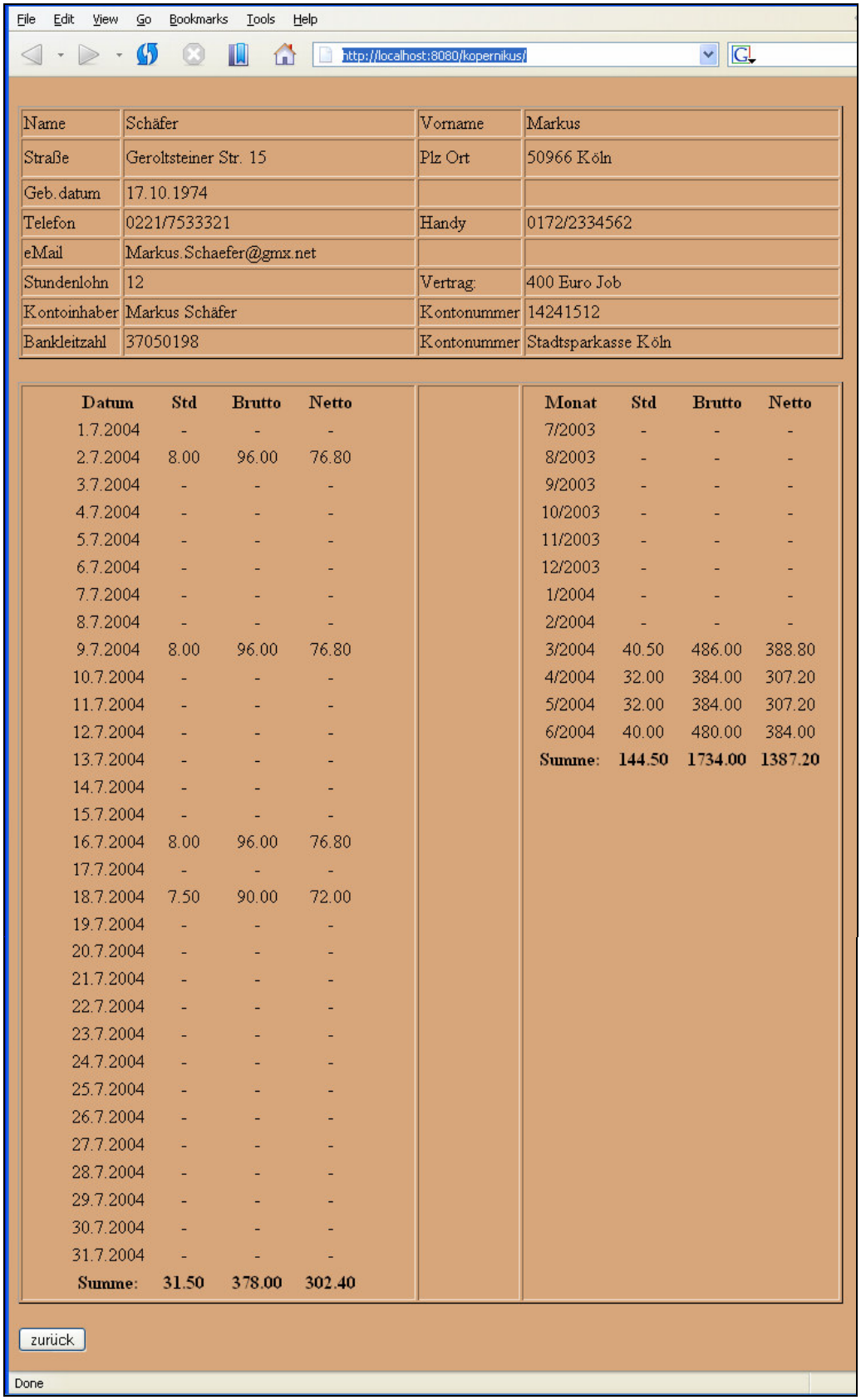
9.1 Anzeigen

Die Anzeige der Mitarbeiter dient zwei Zwecken. Zum Einen kann man sich alle Mitarbeiter anzeigen lassen, welche momentan beim Unternehmen beschäftigt sind. Dies kann entweder der Übersicht am Bildschirm dienen, wenn man beispielsweise einen Mitarbeiter anrufen möchte, kann aber auch dazu genutzt werden, ausgedruckt und als Telefonliste an die Mitarbeiter verteilt zu werden. Damit so keine vertraulichen Daten in unbefugte Hände gelangen, sind in dieser Übersicht ausschließlich die Kontaktdaten dargestellt.



Name	Adresse	Telefon	Handy	Geburtstag
Abels, Stefan	Neumarkt 14, 50939 Köln	0221/444310	0172/4249333	14.12.1977
Bernhardt, Christian	Nord Süd Fahrt 54, 50677 Köln	0221/5634534	0172/3214132	14.4.1976
Ducks, Maike	Ewigstr. 134, 50637 Köln	0221/4876586	0152/5543625	4.1.1984
Dückers, Thomas	Glasigplatz 2, 50559 Köln	0221/3141553	0164/4215435	3.12.1983
Engbers, Frauke	Apostelnst. 33, 50667 Köln	0221/2132142	0161/4324452	26.4.1973
Fischer, Claudia	Hansaring 26, 50668 Köln	0221/5266642	0172/2435455	31.7.1980
Grieger, Frank	Taunusstr. 43, 50622 Köln	0221/452345	0172/4321412	16.4.1972
Höpp, Stefan	Thieboldsgasse 24, 50667 Köln	0221/2142314	0172/2063219	14.12.1977
Lazar, Thomas	Sülzgürtel 21, 50324 Köln	0221/4324235	0177/32142315	21.4.1974
Marl, Karen	Herzogstr. 24, 50345 Köln	0221/324245	0171/3125324	21.1.1972
Mayen, Agnes	Goldsteinstr. 27, 51444 Bergisch Gladbach	02233/4125563	0171/5467326	1.4.1979
Meerfels, Heike	Klettenberggürtel 124, 50939 Köln	0221/9691227		12.6.1981
Rantanplan, Martin	Maarweg 65, 50225 Köln	0221/8768323	0169/3545532	12.12.1979
Rieger, Virginia	Subbelrather Str. 2, 50244 Köln	0221/4254322	0173/62546232	19.4.1979
Schmidt, Matthias	Barthel Str. 32, 50111 Köln	0221/345522	0173/7887325	27.11.1978
Schäfer, Markus	Geroltsteiner Str. 15, 50966 Köln	0221/7533321	0172/2334562	17.10.1974
Siemens, Nadine	Hauptstr. 12, 51004 Hürth	02234/431254	0169/4324321	15.2.1981
Winkler, Matthias	Tunisstr. 35, 50667 Köln	0221/8953982	0172/4329387	16.8.1976

Abbildung 9-1 Mitarbeiterliste



Name	Schäfer	Vorname	Markus
Straße	Geroltsteiner Str. 15	Plz Ort	50966 Köln
Geb.datum	17.10.1974		
Telefon	0221/7533321	Handy	0172/2334562
eMail	Markus.Schaefer@gmx.net		
Stundenlohn	12	Vertrag:	400 Euro Job
Kontoinhaber	Markus Schäfer	Kontonummer	14241512
Bankleitzahl	37050198	Kontonummer	Stadtsparkasse Köln

Datum	Std	Brutto	Netto
1.7.2004	-	-	-
2.7.2004	8.00	96.00	76.80
3.7.2004	-	-	-
4.7.2004	-	-	-
5.7.2004	-	-	-
6.7.2004	-	-	-
7.7.2004	-	-	-
8.7.2004	-	-	-
9.7.2004	8.00	96.00	76.80
10.7.2004	-	-	-
11.7.2004	-	-	-
12.7.2004	-	-	-
13.7.2004	-	-	-
14.7.2004	-	-	-
15.7.2004	-	-	-
16.7.2004	8.00	96.00	76.80
17.7.2004	-	-	-
18.7.2004	7.50	90.00	72.00
19.7.2004	-	-	-
20.7.2004	-	-	-
21.7.2004	-	-	-
22.7.2004	-	-	-
23.7.2004	-	-	-
24.7.2004	-	-	-
25.7.2004	-	-	-
26.7.2004	-	-	-
27.7.2004	-	-	-
28.7.2004	-	-	-
29.7.2004	-	-	-
30.7.2004	-	-	-
31.7.2004	-	-	-
Summe:	**31.50**	**378.00**	**302.40**

Monat	Std	Brutto	Netto
7/2003	-	-	-
8/2003	-	-	-
9/2003	-	-	-
10/2003	-	-	-
11/2003	-	-	-
12/2003	-	-	-
1/2004	-	-	-
2/2004	-	-	-
3/2004	40.50	486.00	388.80
4/2004	32.00	384.00	307.20
5/2004	32.00	384.00	307.20
6/2004	40.00	480.00	384.00
Summe:	**144.50**	**1734.00**	**1387.20**

Abbildung 9-2 Detailansicht eines Mitarbeiters

Zum Anderen gelangt man über diese Seite zur Datailansicht der Mitarbeiter. Diese Seite ist über das Anklicken des Namens des betreffenden Mitarbeiters erreichbar. Über den Link wird der Detailseite die ID des Mitarbeiters übergeben, mit Hilfe derer dann der Mitarbeiter per Mitarbeiter.get(String aID) aus der Datenbank geholt wird.

Über das nun erlangte Mitarbeiter-Objekt gelangt man an die personenbezogenen Daten, welche per entsprechender get-Methode in der Tabelle mit den Kontaktdaten im oberen Bereich ausgegeben werden.

Neben den Kontaktdaten befindet sich auf der Detailseite auch die Übersicht über die geleisteten Arbeitsstunden und die Brutto-, sowie Nettolöhne. Diese Übersicht teilt sich in zwei Spalten. In der linken Seite sieht man eine Übersicht über die einzelnen Tage des jeweils aktuellen Monats und in der rechten Spalte über die letzten zwölf Monate. In jeder Spalte findet sich neben den Einzelwerten auch jeweils die Summe der Stunden, Brutto- und Nettolöhnen.

9.2 Anlegen

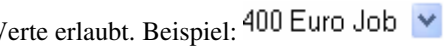
Das Anlegen eines Mitarbeiters ist nötig, jedes Mal, wenn ein neuer Mitarbeiter eingestellt wird. Hier ist dann die Angabe der personenbezogenen Daten, der Vertragsdaten und der Kontodaten nötig.

Die personenbezogenen Daten sind bis auf Ausnahme der Handynummer obligatorisch[23]. Die Angabe der Telefonnummer ist nicht wie die Handynummer obligatorisch, da es sonst sein könnte, dass die Möglichkeit der telefonischen Kontaktaufnahme fehlt.

Die Vertragsdaten setzen sich zusammen aus zwei Dropdown[24]-Feldern. Eines dient zur Auswahl des Brutto-Stundenlohns und ein anderes dient zum Auswählen einer Vertragsart.

---

[23] Es handelt sich bei diesen Feldern also um Pflichtfelder.

[24] Ein Auswahlfeld, welches nach unten aufklappt und die Auswahl aus einer Menge statischer Werte erlaubt. Beispiel: 

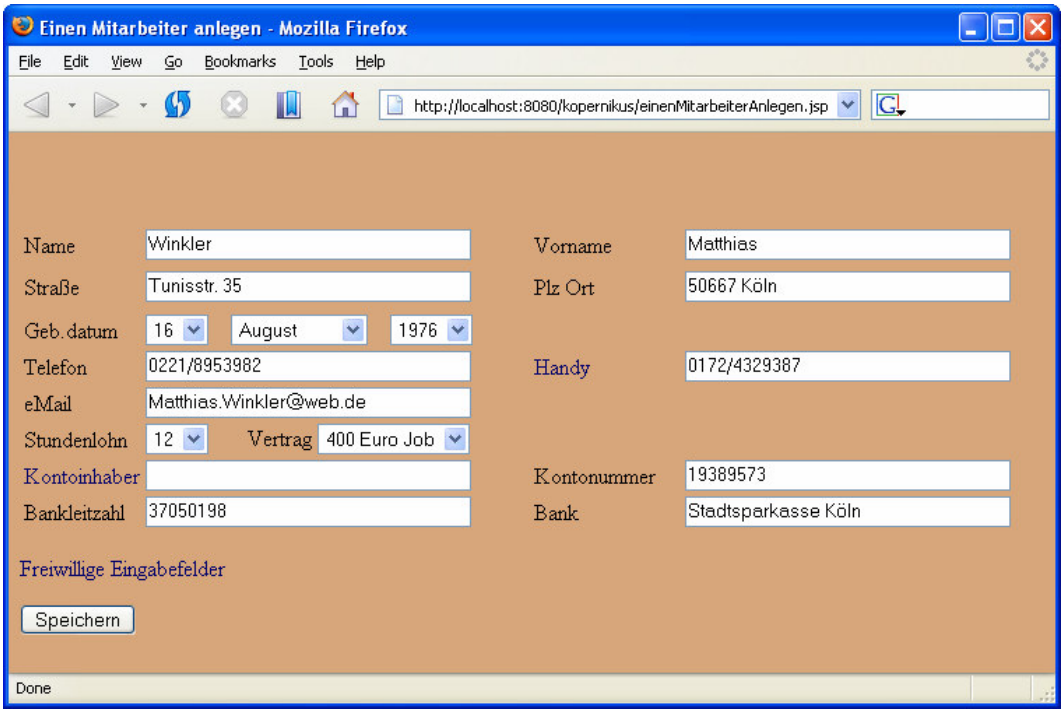


Abbildung 9-3 Anlegen eines Mitarbeiters

9.3 Ändern

Neben der Möglichkeit neue Mitarbeiter anzulegen, muss es noch die Möglichkeit geben, bestehende Mitarbeiterinformationen zu ändern. So ist es beispielsweise denkbar, dass ein Mitarbeiter umzieht und so natürlich seine Adresse angepasst werden muss. Es kann aber auch sein, dass ein Mitarbeiter seine Vertragsart ändern möchte. Dies ist jederzeit auch im laufenden Monat möglich, da die Klasse Mitarbeiter so implementiert ist, dass sie trotzdem am Ende des Monats entsprechend den richtigen Lohn berechnet hat. Dies wird dadurch erreicht, dass der entsprechende Tageslohn jeden Tag durch die Freigabe des Tages konkret berechnet und versteuert wird.

Der Änderungsdialog sieht, damit der Benutzer eine ihm vertraute und gewohnte Oberfläche vorfindet, auf der er sich sofort orientieren kann, wie der Dialog für das Anlegen eines neuen Mitarbeiters (siehe Abbildung 9-3) aus. Der einzige Unterschied besteht hier darin, dass die Werte für alle Felder schon mit den momentan in der Datenbank stehenden Werte gefüllt sind. Nach den Änderungen und der Bestätigung durch den Benutzer, überprüft die Applikation, ob

Informationen geändert wurden und falls dies der Fall ist, führt sie die entsprechenden Änderungen an der Datenbank durch.

9.4 Löschen

Das Löschen eines Mitarbeiters, was dem Vorgang der Entlassung eines Mitarbeiters entspricht, sollte man meinen, wäre vollkommen trivial[25]. Dies ist jedoch nicht uneingeschränkt der Fall. So verfügt der Mitarbeiter über eine ID, welche in einigen Statistik- und Abrechnungstabellen indiziert wird. Das bedeutet, dass über diese ID eine Verknüpfung zum Mitarbeiter hergestellt wird, über den, wenn man ihm folgt, man an beispielsweise den Stundenlohn gelangt um die Abendausgaben zu berechnen. Wenn nun der Mitarbeiter vor Ende des Monats und somit vor dem Monatsabschluss entlassen wird, läuft die Berechnung der Bilanz ins Leere, da sie einen Lohnposten hat, welcher die Informationen eines nicht mehr vorhandenen Mitarbeiters benötigt.

Um dieses Problem (die referenetiellen Integrität) einfach und wirkungsvoll zu umgehen und zu lösen, unterscheidet die Applikation zwei Arten von Mitarbeiter, aktive und passive. Der aktive Mitarbeiter ist ein aktuell Beschäftigter, der auch in allen Buchungsdialogen erscheint und somit buchbar ist. Neben diesem aktiven Mitarbeiter existieren noch die passiven Mitarbeiter. Diese Mitarbeiter sind nicht buchbar und erscheinen auch in keinerlei Auswahl oder Übersichten (inklusive der Telefonliste) mehr. Sie sind ausschließlich als Kontaktdatensatz in der Datenbank vorhanden.

[25] bezeichnet einen sehr einfachen Prozess

10 Modul Mitarbeitermanagement

Dieses Modul besteht im Wesentlichen aus der Funktion des Personal-, beziehungsweise Schichtplans.

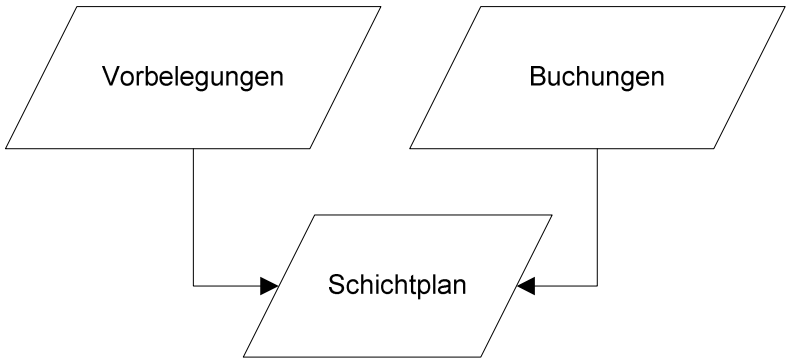
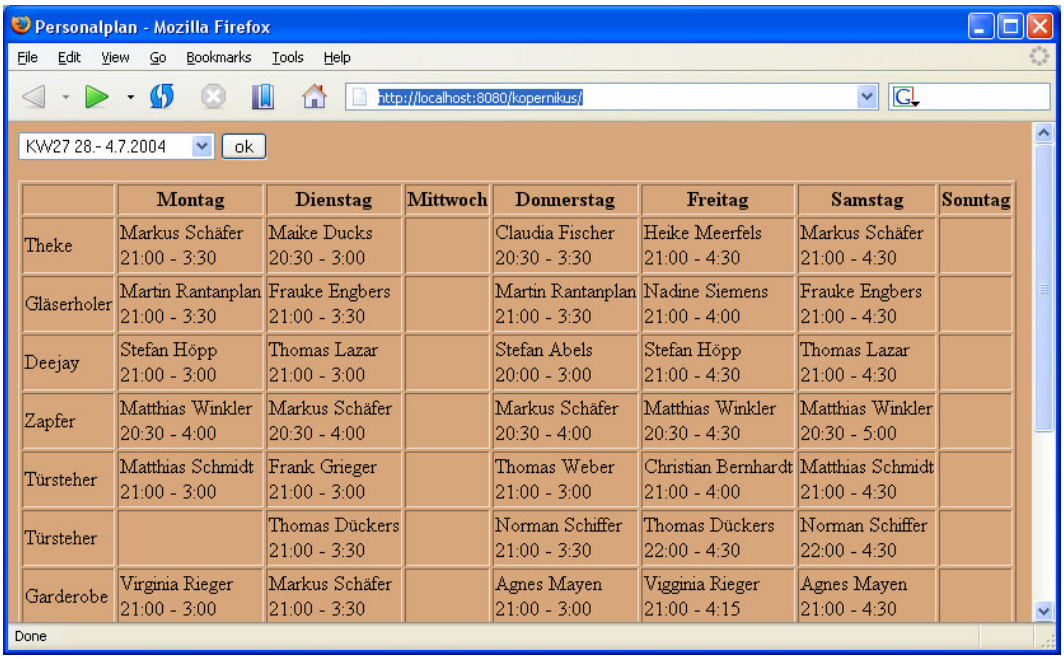


Abbildung 10-1 Erstellung des Schichtplans

10.1 Anzeigen

Hier wird der Schichtplan für den aktuellen Monat angezeigt. Die Seite ist so konstruiert, dass sie sich passend auf ein Querformat A4 Blatt drucken lässt. Über ein Auswahlfeld lässt sich auswählen, welche Woche angezeigt werden soll. So lassen sich auch alte Schichtpläne nachträglich einsehen.

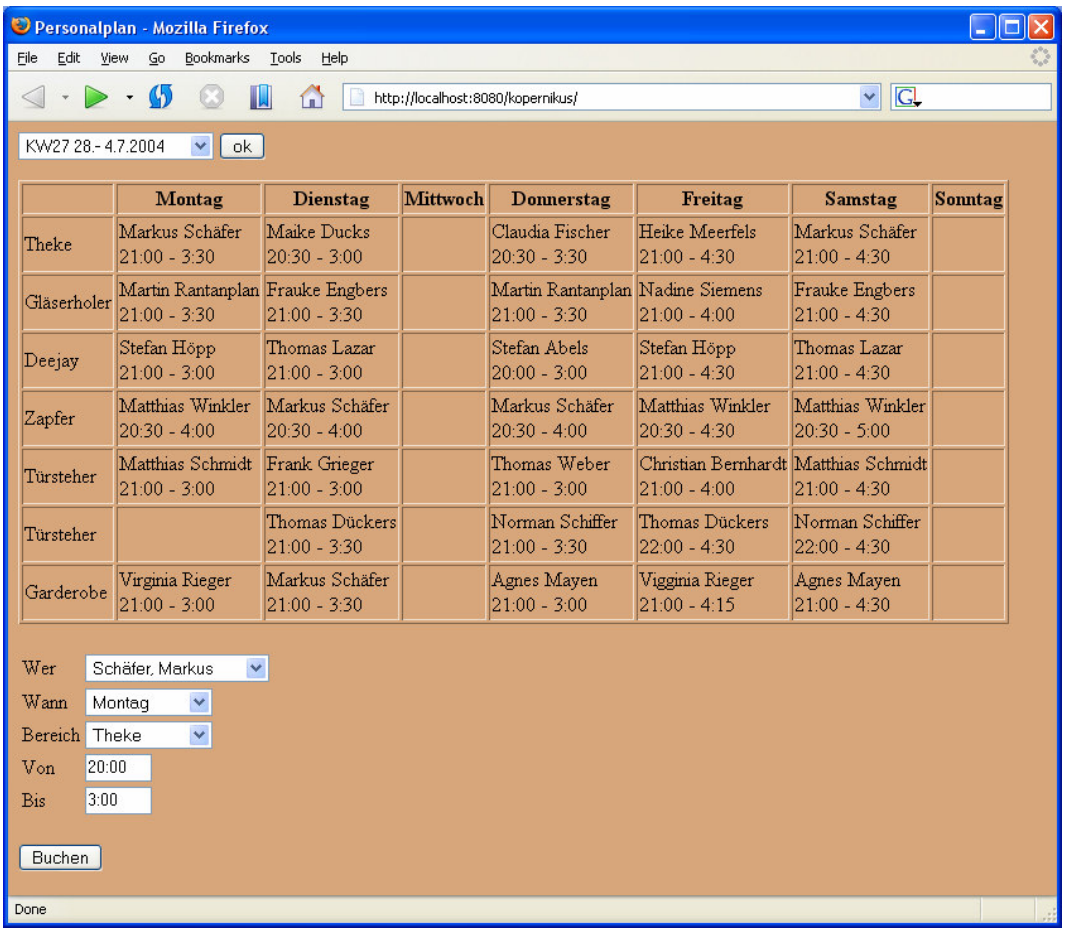


	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
Theke	Markus Schäfer 21:00 - 3:30	Maike Ducks 20:30 - 3:00		Claudia Fischer 20:30 - 3:30	Heike Meerfels 21:00 - 4:30	Markus Schäfer 21:00 - 4:30	
Gläserholer	Martin Rantanplan 21:00 - 3:30	Frauke Engbers 21:00 - 3:30		Martin Rantanplan 21:00 - 3:30	Nadine Siemens 21:00 - 4:00	Frauke Engbers 21:00 - 4:30	
Deejay	Stefan Höpp 21:00 - 3:00	Thomas Lazar 21:00 - 3:00		Stefan Abels 20:00 - 3:00	Stefan Höpp 21:00 - 4:30	Thomas Lazar 21:00 - 4:30	
Zapfer	Matthias Winkler 20:30 - 4:00	Markus Schäfer 20:30 - 4:00		Markus Schäfer 20:30 - 4:00	Matthias Winkler 20:30 - 4:30	Matthias Winkler 20:30 - 5:00	
Türsteher	Matthias Schmidt 21:00 - 3:00	Frank Grieger 21:00 - 3:00		Thomas Weber 21:00 - 3:00	Christian Bernhardt 21:00 - 4:00	Matthias Schmidt 21:00 - 4:30	
Türsteher		Thomas Dückers 21:00 - 3:30		Norman Schiffer 21:00 - 3:30	Thomas Dückers 22:00 - 4:30	Norman Schiffer 22:00 - 4:30	
Garderobe	Virginia Rieger 21:00 - 3:00	Markus Schäfer 21:00 - 3:30		Agnes Mayen 21:00 - 3:00	Vigginia Rieger 21:00 - 4:15	Agnes Mayen 21:00 - 4:30	

Abbildung 10-2 Den Personalplan anzeigen

10.2 Buchen

Mit Hilfe dieser Seite kann man Personal buchen, welches an dem entprechenden Tag nicht schon per Vorbelegung arbeitet. Hierfür zeigt sich auf der Seite zunächst der Schichtplan der aktuellen Woche, welche sich aber per Auswahlfeld auf eine der kommenden Wochen verschieben lässt.



	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
Theke	Markus Schäfer 21:00 - 3:30	Maike Ducks 20:30 - 3:00		Claudia Fischer 20:30 - 3:30	Heike Meerfels 21:00 - 4:30	Markus Schäfer 21:00 - 4:30	
Gläserholer	Martin Rantanplan 21:00 - 3:30	Frauke Engbers 21:00 - 3:30		Martin Rantanplan 21:00 - 3:30	Nadine Siemens 21:00 - 4:00	Frauke Engbers 21:00 - 4:30	
Deejay	Stefan Höpp 21:00 - 3:00	Thomas Lazar 21:00 - 3:00		Stefan Abels 20:00 - 3:00	Stefan Höpp 21:00 - 4:30	Thomas Lazar 21:00 - 4:30	
Zapfer	Matthias Winkler 20:30 - 4:00	Markus Schäfer 20:30 - 4:00		Markus Schäfer 20:30 - 4:00	Matthias Winkler 20:30 - 4:30	Matthias Winkler 20:30 - 5:00	
Türsteher	Matthias Schmidt 21:00 - 3:00	Frank Grieger 21:00 - 3:00		Thomas Weber 21:00 - 3:00	Christian Bernhardt 21:00 - 4:00	Matthias Schmidt 21:00 - 4:30	
Türsteher		Thomas Dückers 21:00 - 3:30		Norman Schiffer 21:00 - 3:30	Thomas Dückers 22:00 - 4:30	Norman Schiffer 22:00 - 4:30	
Garderobe	Virginia Rieger 21:00 - 3:00	Markus Schäfer 21:00 - 3:30		Agnes Mayen 21:00 - 3:00	Vigginia Rieger 21:00 - 4:15	Agnes Mayen 21:00 - 4:30	

Abbildung 10-3 Einen Mitarbeiter buchen

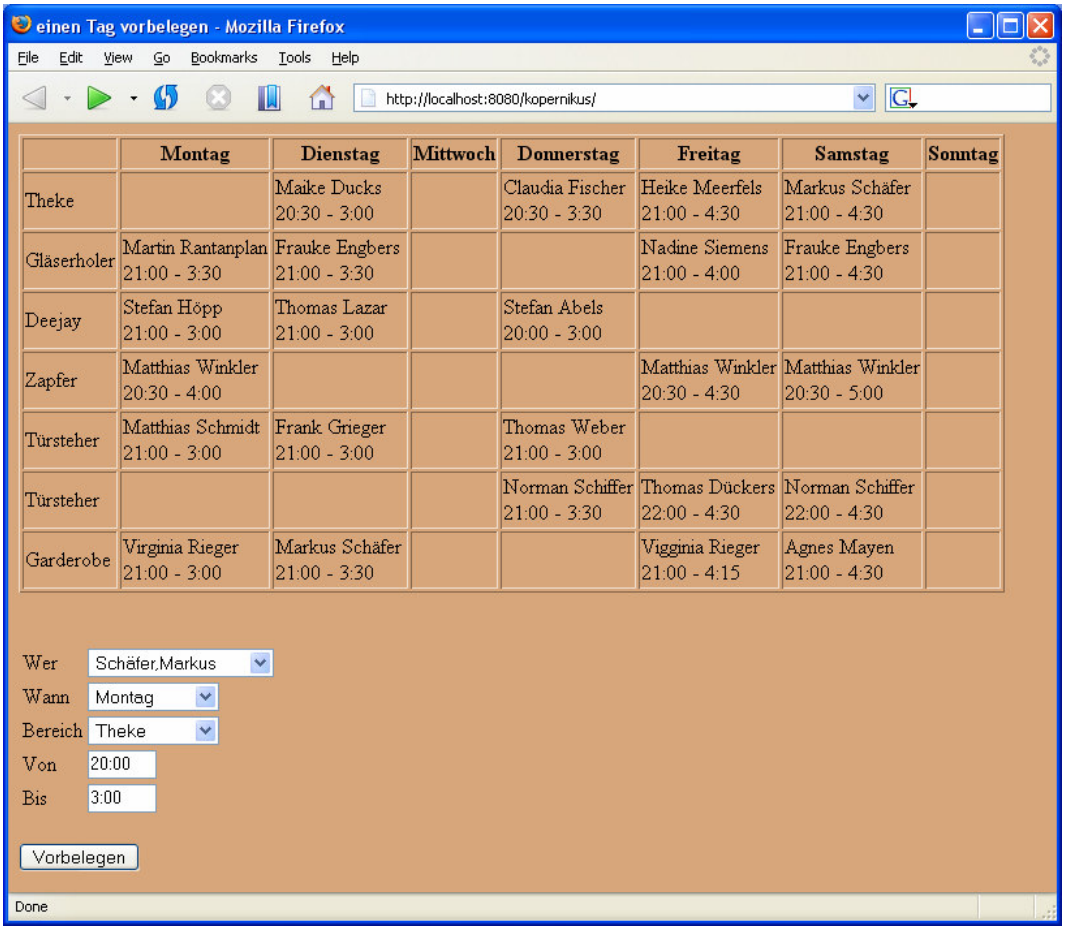
Wenn man nun einen Mitarbeiter für einen Tag buchen möchte, dann wählt man zunächst den Mitarbeiter aus und entscheidet sich dann für Wochentag, wo er arbeiten soll und wann. Mitarbeiter sind nicht auf ihre speziellen Arbeitsbereiche festgelegt. Das heißt, dass Thekenpersonal durchaus auch als Gläserholer arbeiten kann. Über die Eignung hierfür entscheidet der Abendverantwortliche.

10.3 Löschen

Hier lässt sich über ein Drop-Down-Menü, in dem die gebuchten Mitarbeiter für die selektierte Woche stehen, ein Mitarbeiter auswählen und löschen. Bevor jedoch ein Mitarbeiter endgültig aus dem Schichtplan gelöscht wird, muss der Benutzer dies erst noch wiederholt bestätigen.

10.4 Vorbelegung anlegen

Hier sieht man in der oberen Hälfte den momentane Verbelegung der einzelnen Wochentage. Im Gegensatz zu den anderen Dialogen ist hier keine Auswahl der Woche möglich (und nötig), da diese Seite das Muster jeder zukünftigen Woche vorgibt.



	Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag
Theke		Maike Ducks 20:30 - 3:00		Claudia Fischer 20:30 - 3:30	Heike Meerfels 21:00 - 4:30	Markus Schäfer 21:00 - 4:30	
Gläserholer	Martin Rantanplan 21:00 - 3:30	Frauke Engbers 21:00 - 3:30			Nadine Siemens 21:00 - 4:00	Frauke Engbers 21:00 - 4:30	
Deejay	Stefan Höpp 21:00 - 3:00	Thomas Lazar 21:00 - 3:00		Stefan Abels 20:00 - 3:00			
Zapfer	Matthias Winkler 20:30 - 4:00				Matthias Winkler 20:30 - 4:30	Matthias Winkler 20:30 - 5:00	
Türsteher	Matthias Schmidt 21:00 - 3:00	Frank Grieger 21:00 - 3:00		Thomas Weber 21:00 - 3:00			
Türsteher				Norman Schiffer 21:00 - 3:30	Thomas Dückers 22:00 - 4:30	Norman Schiffer 22:00 - 4:30	
Garderobe	Virginia Rieger 21:00 - 3:00	Markus Schäfer 21:00 - 3:30			Vigginia Rieger 21:00 - 4:15	Agnes Mayen 21:00 - 4:30	

Abbildung 10-4 Einen Tag vorbelegen

Um einen Tag für einen Mitarbeiter vorzubelegen, wählt man den Mitarbeiter, den Tag, den Bereich und die Uhrzeit aus, für die er eingeteilt wird, regulär wöchentlich zu arbeiten.

Nach dem Klicken auf ok belgt die Applikation die folgenden neun Wochen automatisch vor. Diese Vorbelgung wird automatisch jeden Tag von der Applikation um einen Tag weitergeführt (siehe Kapitel 8).

10.5 Vorbelegung anzeigen

Mit Hilfe dieser Funktion bekommt man eine Übersicht über die aktuell vorbelegten Tage. Bei dieser Darstellung handelt es sich im Grunde um die Tabelle aus Abbildung 10-4.

10.6 Vorbelegung löschen

Um eine Vorbelegung löschen zu können, muss man hier zunächst in einem Dropdownmenu die zu löschende Vorbelegung auswählen. Wenn man das Löschen nun bestätigt hat, werden die entsprechenden Vorbelegungen der nächsten neun Wochen gelöscht.

11 Modul Abrechnung

11.1 Grundsätzliches

In allen hier vorkommenden Funktionen werden die zuvor errechneten und in der Datenbank nach der Tagesfreigabe (sie Kapitel XX) gespeicherten Brutto- und Nettolöhne verwendet.

Die Berechnung der Nettolöhne ist mit einem sehr hohem Aufwand verbunden. Es ist nicht so, dass einfach nur ein fester Prozentsatz von einem Lohn abgezogen werden muss, sondern es gibt eine Menge denkbarer Verträge. Jeder dieser Verträge setzt sich aus unterschiedlichen Steuern zusammen, welche sowohl den Arbeitnehmer betreffen können, als auch zu Lasten des Arbeitgebers oder beiden gehen können.

Die Besteuerung eines Vertrages wird im Grunde in drei Tabellen verwaltet. Als erstes gibt es da die Tabelle, welche sämtliche Verträge beinhaltet. Dann gibt es eine Tabelle, welche alle Arten von Steuern verzeichnet, mit denen die einzelnen Verträge belastet werden und dann gibt es da noch die Tabelle Steuern, welche die einzelnen Steuern beschreibt. In dieser Beschreibung steht drin, wer den Steueranteil bezahlen muss und wie viel er jeweils bezahlen muss. Diese Prozentwerte bilden dann die Grundlage zur Versteuerung der Einkommen.

Vertragsname	**Beschreibung**
400 Euro Job	Ein Vertrag für einen 400 Euro Job
Nebenjob	Ein Vertrag für einen normalen Nebenjob

Tabelle 1 Table Vertrag

Vertragsname	**Besteuerung**
400 Euro Job	Sozialversicherung 400 Euro Arbeitnehmer
400 Euro Job	Sozialversicherung 400 Euro Arbeitgeber
Nebenjob	Sozialversicherung
Nebenjob	Rente
Nebenjob	Krankenversicherung Arbeitnehmer
Nebenjob	Krankenversicherung Arbeitgeber

Tabelle 2 Table Besteuerung

Name	WerZahlt	Wieviel
Sozialversicherung 400 Euro Arbeitnehmer	Arbeitnehmer	10.00
Sozialversicherung 400 Euro Arbeitgeber	Arbeitgeber	10.00
Sozialversicherung	Arbeitnehmer	34.00
Rente	Arbeitnehmer	15.00
Krankenversicherung Arbeitnehmer	Arbeitnehmer	7.50
Krankenversicherung Arbeitgeber	Arbeitgeber	7.50

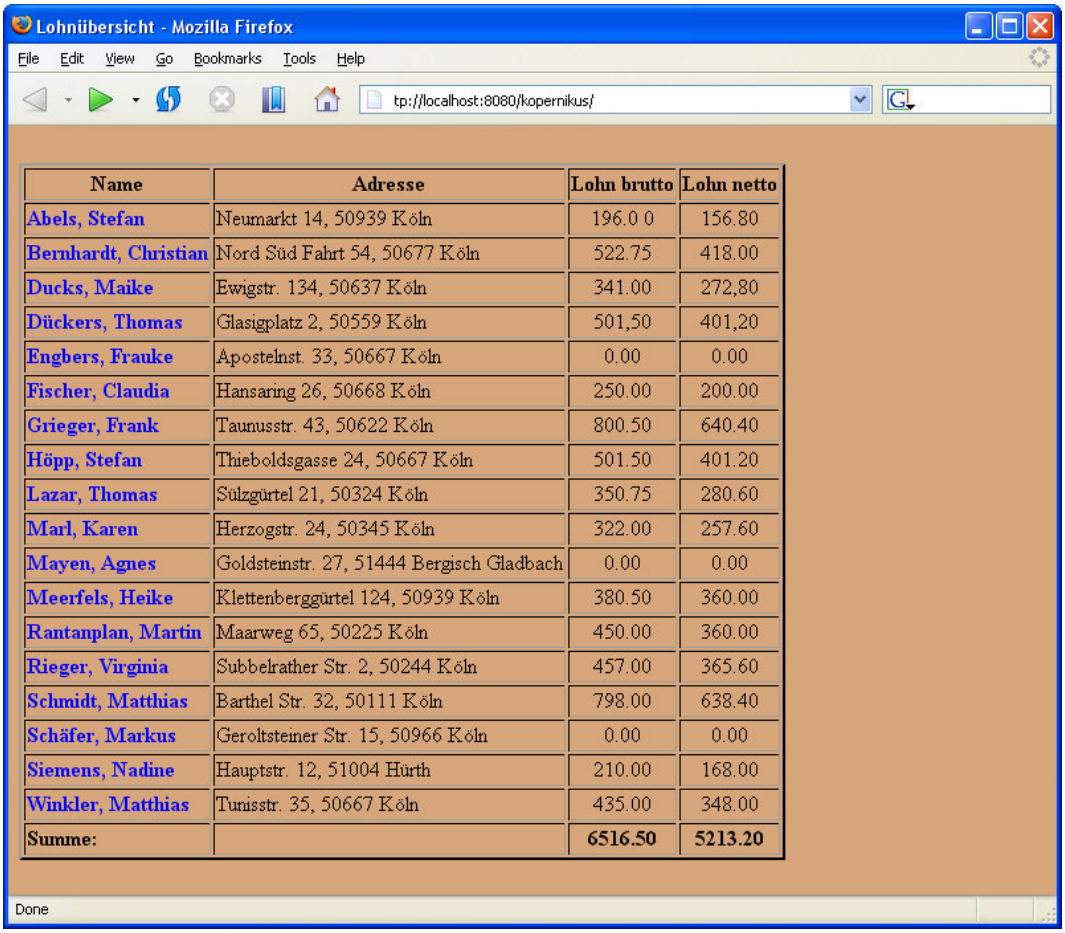
Tabelle 3 Table Steuern

Neben den Steuern, welche die Mitarbeiter betreffen, gibt es zusätzlich die einnahmenbelastende Mehrwertsteuer[26] auf Artikel, welche verkauft werden. Diese Steuer wird beim Freigeben eines Tages berechnet und in der Tabelle Einnahmenkonto für jeden Artikeldatensatz mitgeschrieben. Nach dem Freigeben eines Monats müssen diese Steuern natürlich gesammelt werden und die Überweisung an das Finanzamt eingeleitet werden. Unnötig zu Erwähnen, dass diese Funktionen allerhöchste Genauigkeit erfordern und sich schon Abweichungen im Centbereich verbieten. Gerade diese kleinen Abweichungen bleiben oft unerkannt und können sich die Menge der Mitarbeiter und übers Jahr gesehen erheblich summieren. So summieren sich zum Beispiel 10 Cent pro Mitarbeiter und Abend die nicht weiter auffallen, bei pro Abend arbeitenden 17 Personen und 280 Arbeitstagen auf einen Fehlbetrag von 476 Euro, den die Steuerbehörden erst einmal gerechtfertigt sehen wollen. Dies gilt natürlich nicht bloß für die Mehrwertsteuer, sondern insbesondere auch für die Lohnsteuer, Sozialabgaben, usw.

11.2 Lohnübersicht

Die Lohnübersicht zeigt zunächst, wie in Abbildung 11-1 zu sehen, alle Mitarbeiter mit ihren im aktuellen Monat verdientem Brutto-Lohn und Netto-Lohn an. Neben dieser Übersicht gibt es jedoch noch eine Detailansicht (siehe Abbildung 9-2), zu der man durch ein Klicken auf den entsprechenden Mitarbeiter gelangt.

[26] Die Mehrwertsteuer beträgt in Deutschland 16%. Neben dieser gibt es noch eine ermäßigte Mehrwertsteuer, welche auf bestimmte Artikel angewendet wird, welche nur 7% des Nettopreises beträgt.



Name	Adresse	Lohn brutto	Lohn netto
Abels, Stefan	Neumarkt 14, 50939 Köln	196.0 0	156.80
Bernhardt, Christian	Nord Süd Fahrt 54, 50677 Köln	522.75	418.00
Ducks, Maike	Ewigstr. 134, 50637 Köln	341.00	272,80
Dückers, Thomas	Glasigplatz 2, 50559 Köln	501,50	401,20
Engbers, Frauke	Apostelnst. 33, 50667 Köln	0.00	0.00
Fischer, Claudia	Hansaring 26, 50668 Köln	250.00	200.00
Grieger, Frank	Taunusstr. 43, 50622 Köln	800.50	640.40
Höpp, Stefan	Thieboldsgasse 24, 50667 Köln	501.50	401.20
Lazar, Thomas	Sülzgürtel 21, 50324 Köln	350.75	280.60
Marl, Karen	Herzogstr. 24, 50345 Köln	322.00	257.60
Mayen, Agnes	Goldsteinstr. 27, 51444 Bergisch Gladbach	0.00	0.00
Meerfels, Heike	Klettenberggürtel 124, 50939 Köln	380.50	360.00
Rantanplan, Martin	Maarweg 65, 50225 Köln	450.00	360.00
Rieger, Virginia	Subbelrather Str. 2, 50244 Köln	457.00	365.60
Schmidt, Matthias	Barthel Str. 32, 50111 Köln	798.00	638.40
Schäfer, Markus	Geroltsteiner Str. 15, 50966 Köln	0.00	0.00
Siemens, Nadine	Hauptstr. 12, 51004 Hürth	210.00	168.00
Winkler, Matthias	Tunisstr. 35, 50667 Köln	435.00	348.00
Summe:		**6516.50**	**5213.20**

Abbildung 11-1 Lohnübersicht

In der Übersicht sieht man zunächst alle mitarbeiterbezogenen Daten. Dies sind zum Beispiel das Geburtsdatum, Vertragsart oder die Kontoverbindung. Desweiteren zeigt sich dort die tageweise und als Summe aufgeschlüsselten geleisteten Arbeitsstunden, Netto-Löhne und Brutto-Löhne. Neben der tageweisen Aufschlüsselung existiert noch eine monateweise Aufschlüsselung. Hier sind ebenfalls die geleisteten Arbeitstunden, Netto-Löhne und Bruttolöhne verzeichnet, nur halt für die Monate, statt die Tage.

11.3 Abend anzeigen

Diese Funktion zeigt einen einzelnen Abend tabellarisch an, um die Abenddabrechnung vornehmen zu können. Er zeigt also im Grunde die Ergebnisse der klassischen[27] Abendabrechnung an. Hier ist also eine Bilanz des Abends mit ihren einzelnen Berechnungsdetails zu finden.

[27] damit ist die aktuelle IST-Abrechnung (siehe Kapitel 2.4) gemeint.

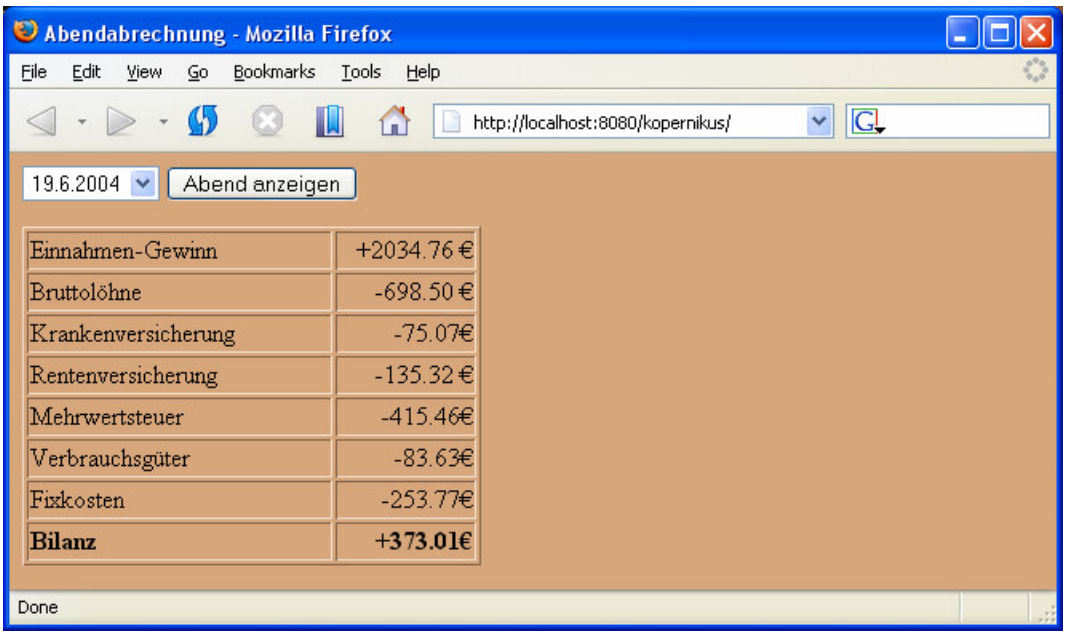


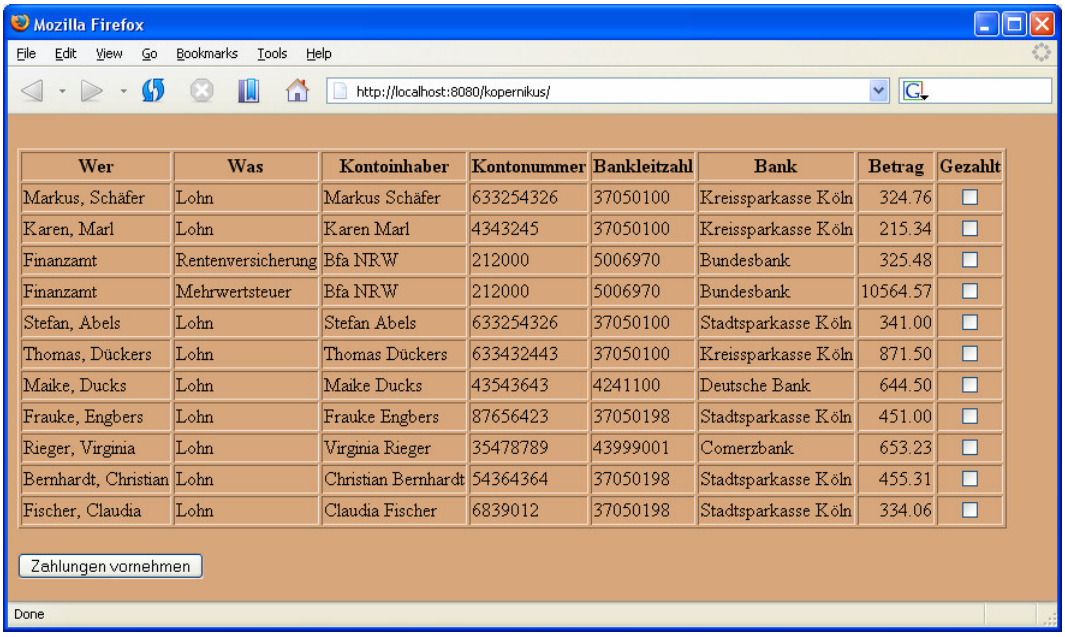
Abbildung 11-2 Abendabrechnung anzeigen

11.4 Auszahlungen

Hier werden die Auszahlungen angezeigt die für den ausgewählten Monat getätigt werden müssen. Dies sind z.B. die zu entrichtenden Steuern und Löhne.
Zunächst wählt man den Monat aus, zu dem man die vorzunehmenden Auszahlungen sehen möchte. Die Applikation ist so geschrieben, dass sie hier nur eine Auswahl der Monate zulässt, welche freigegeben[28] sind. Die freigegebenen Monate sind hier zur besseren Visualisierung farblich hervorgehoben. Die freigegebenen Monat sind hierfür grün und die noch nicht freigegebenen rot.

Die nun folgende Seite zeigt für den gewählten Monat eine Tabelle an. In dieser Tabelle sind die zu zahlenden Beträge verzeichnet. Zusätzlich finden sich dort noch die Informationen für wen, was und wohin zu überweisen ist. Die Posten, die man erledigt, das heißt überwiesen oder anderweitig ausgezahlt hat, markiert man mit einem Häkchen am Ende der Zeile. Nach einem Bestätigen aller gesetzten Häkchen durch ein Klicken auf das OK, werden die entsprechenden Änderungen an der Datenbank vorgenommen.

[28] ein Monat ist freigegeben, wenn er im dazugehörigen Dialog als abgeschlossen markiert ist.



Wer	Was	Kontoinhaber	Kontonummer	Bankleitzahl	Bank	Betrag	Gezahlt
Markus, Schäfer	Lohn	Markus Schäfer	633254326	37050100	Kreissparkasse Köln	324.76	☐
Karen, Marl	Lohn	Karen Marl	4343245	37050100	Kreissparkasse Köln	215.34	☐
Finanzamt	Rentenversicherung	Bfa NRW	212000	5006970	Bundesbank	325.48	☐
Finanzamt	Mehrwertsteuer	Bfa NRW	212000	5006970	Bundesbank	10564.57	☐
Stefan, Abels	Lohn	Stefan Abels	633254326	37050100	Stadtsparkasse Köln	341.00	☐
Thomas, Dückers	Lohn	Thomas Dückers	633432443	37050100	Kreissparkasse Köln	871.50	☐
Maike, Ducks	Lohn	Maike Ducks	43543643	4241100	Deutsche Bank	644.50	☐
Frauke, Engbers	Lohn	Frauke Engbers	87656423	37050198	Stadtsparkasse Köln	451.00	☐
Rieger, Virginia	Lohn	Virginia Rieger	35478789	43999001	Comerzbank	653.23	☐
Bernhardt, Christian	Lohn	Christian Bernhardt	54364364	37050198	Stadtsparkasse Köln	455.31	☐
Fischer, Claudia	Lohn	Claudia Fischer	6839012	37050198	Stadtsparkasse Köln	334.06	☐

Abbildung 11-3 Ausgabe der zu tätigenden Überweisungen

11.5 Tabellarisch

Die tabellarische Abrechnug zeigt die Abrechnungen über einen gewählten Zeitraum tabellarisch an. In dieser Tabelle sind die einelnen Lohnnebenkosten, sowie die restlichen Einnahmen und Ausgaben als Summe verzeichnet.

11.6 Grafisch

[Ein Bild sagt mehr als tausend Worte]

Um die Bilanzen über einen Zeitraum grafisch auszuwerten, wählt man zunächst den Zeitraum aus, welcher ausgertet werden soll. Dieser Zeitraum wird dann an ein Applet (Auszug, siehe Abbildung 11-5) übergeben, welches sich um die Visualisierung kümmert. In der Darstellung zeigen sich die beiden Graphen für Umsatz und Gewinn.

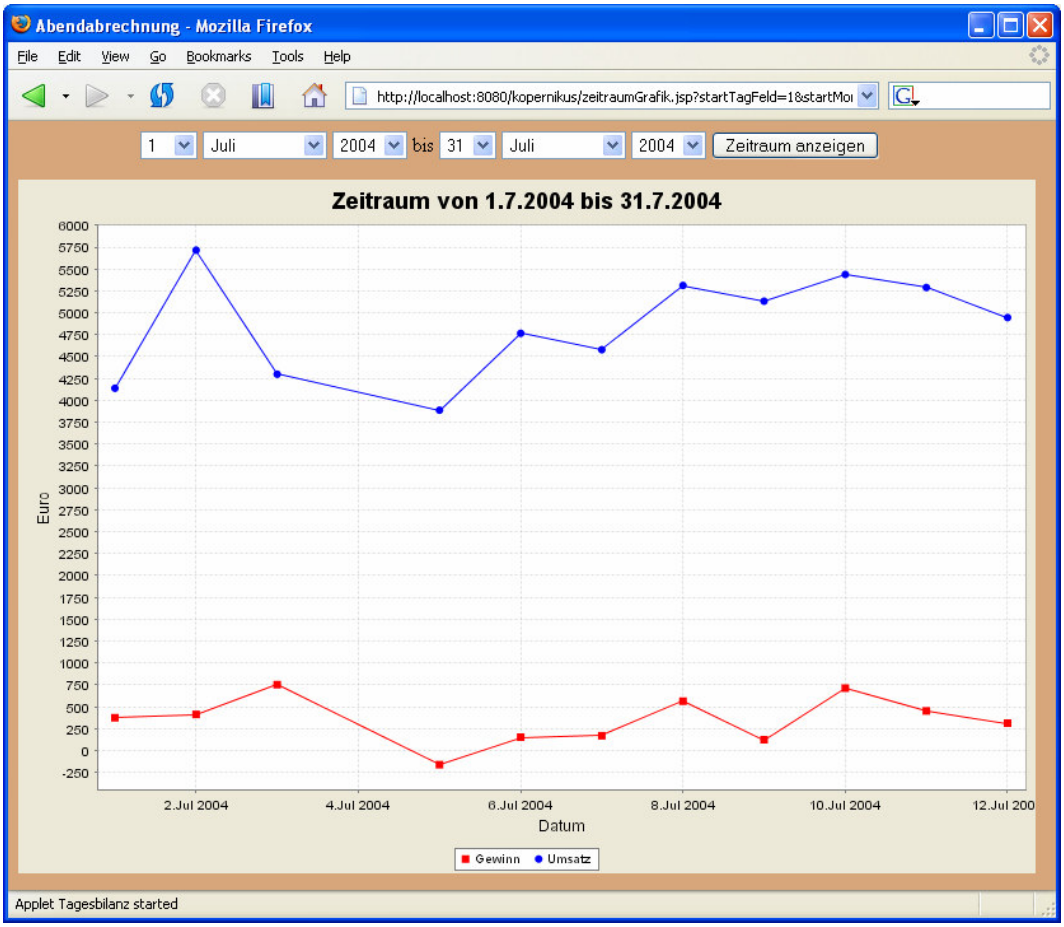


Abbildung 11-4 Grafische Auswertung der Umsätze und Bilanzen

11.6.1 jFreeChart

jFreeChart ist ein Open-Source Projekt unter der Gnu Lesser Public Licence[29] (vgl. [Lgpl]) zur Visualisierung von Diagrammen. Man kann damit beispielsweise Balken, Kreis oder Liniendiagramme darstellen.

In Abbildung 11-5 sieht man ein Code-Beispiel für ein mit jFreeChart erzeugtes Liniendiagramm. Um ein Liniendiagramm erstellen zu können muss man über eine Reihe von Wertepaaren verfügen. Zunächst muss eine TimeSeries angelegt werden. Eine TimeSeries muss für jeden Reihe von Wertepaaren erstellt und gefüllt werden. Die TimeSeries bekommt beim Instatiieren den Namen für die Legende und die Skaleneinteilung übergeben. Die Skaleneinteilung kann zum

[29] Die Gnu Lesser Public Licence ist eine Open Source Lizenz, welche nicht ganz so restriktiv wie die Gnu Public Licence ausgelegt ist. Hier ist es zum Beispiel nicht nötig, den Code des verwendenden Programms in jedem Fall offen zu legen.

Beispiel per Tag(Day.class), Monat(Month.Class) oder Jahr(Year.Class) vorgenommen werden. Das Füllen der TimeSeries erfolgt dann einfach durch das Hinzufügen der einzelnen Datum-Wert-Paaren. Wenn man nun alle TimeSeries, die man darstellen will, erzeugt und gefüllt hat, legt man eine TimeSeriesCollection an, welche die TimeSeries sammelt.

Nun kann man das eigentliche chart[30] anlegen. Bei der Instantiierung werden eine Reihe von Parametern übergeben. Hier eine tabellarisch Übersicht über die wichtigsten Parameter[31]:

Nr	Was	Beispiel
1	Der Titel des Diagramms	Zeitraum: 14.2.2004 – 14.4.2004
2	Beschriftung X-Achse	Datum
3	Beschriftung Y-Achse	Euro
4	Die TimeSeriesCollection, welche dargestellt werden soll	dataset

Tabelle 4 Parameter beim Anlegen eines Chart

Die weiteren Parameter, sowie die optionalen Einstellungen, die ergänzend vorgenommen werden, sind in [ChartDoku] dokumentiert. Interessant hier noch zu erwähnen, wäre vielleicht die Zeile:
axis.setDateFormatOverride(**new** SimpleDateFormat("d.MMM yyyy"));
Sie bestimmt, wie das Datumsformat an der X-Achse aussieht. Die Formatierung richtet sich hier an die gängige Konvention. Das bedeutet beispielsweise:

M	1,2,3,...12
MM	01,02,03,...12
MMM	Januar,Februar,März,...,Dezember

Tabelle 5 Datumsformate bei Charts

```
import org.jfree.chart.*

public class Tagesbilanz extends JApplet {
  String startTag = getParameter("startTag");
  String endeTag = getParameter("endeTag");
  CalendarDate vonCalendar = new CalendarDate(startTag);
  CalendarDate bisCalendar = new CalendarDate(endeTag);
  Period per1 = new Period(vonCalendar, bisCalendar);
```

[30] der englische Begriff für Diagramm

[31] detaillierte Beschreibung siehe [ChartDoku]

```
public void start(){
  TimeSeries s1 = new TimeSeries("Gewinn",Day.class);
  ArrayList tmpEinnahmenList = Tageszahlen.get(per1,"Gewinn",
                                        "Arbeitgeber","Einnahme");
  Iterator einnahmen = tmpEinnahmenList.iterator();
  while(einnahmen.hasNext()){
    Tageszahlen tmpTageszahlen = (Tageszahlen)einnahmen.next();
    s1.add(new Day(tmpTageszahlen.getDatum().getDate()),
                  tmpTageszahlen.getWieviel());
  }
  TimeSeries s2 = new TimeSeries("Umsatz",Day.class);
  ArrayList gesamtUmsatzList = Tageszahlen.get(per1,"Umsatz",
                                        "Arbeitgeber","Einnahme");
  Iterator gesamtUmsatz = gesamtUmsatzList.iterator();
  while(gesamtUmsatz.hasNext()){
    Tageszahlen tmpTageszahlen = (Tageszahlen)gesamtUmsatz.next();
    s2.add(new Day(tmpTageszahlen.getDatum().getDate()),
                                    tmpTageszahlen.getWieviel());
  }
  TimeSeriesCollection dataset = new TimeSeriesCollection();
    dataset.addSeries(s1);dataset.addSeries(s2);
  JFreeChart chart = ChartFactory.createTimeSeriesChart(
    "Zeitraum von " + startTag + " bis " + endeTag,
    "Datum","Euro",dataset,true,true,false);
  StandardLegend sl = (StandardLegend) chart.getLegend();
  sl.setDisplaySeriesShapes(true);
  XYPlot plot = chart.getXYPlot();
  XYItemRenderer renderer = plot.getRenderer();
  if (renderer instanceof StandardXYItemRenderer) {
    StandardXYItemRenderer rr = (StandardXYItemRenderer) renderer;
    rr.setPlotShapes(true);rr.setShapesFilled(true);
  }
  DateAxis axis = (DateAxis) plot.getDomainAxis();
  axis.setDateFormatOverride(new SimpleDateFormat("d.MMM yyyy"));
  ChartPanel chartPanel = new ChartPanel(chart);
  chartPanel.setPreferredSize(new java.awt.Dimension(800, 600));
  chartPanel.setMouseZoomable(true, false);
  chartPanel.setPopupMenu(null);
  getContentPane().add(chartPanel);
}
}
```

Abbildung 11-5 Code-Beispiel eines Liniendiagramms

11.7 Freigabe

Die Applikation verfügt über ein mehrstufiges Freigabesystem, welches die Vollständig- und Gültigkeit von Buchungen und Transaktionen verwaltet. Dabei werden die Werte von links nach rechts konkreter. Zum Beispiel werden die Arbeitszeiten im Booking nur abstrakt als „von“ und „bis“ gespeichert und im Arbeitskonto ist für diese Werte bereits der konkrete Lohn ausgerechnet. Andererseits verlieren die Daten von links nach rechts auch an Details. So stehen beispielsweise für jede Theke und jeden Artikel in Einnahmen die Werte noch

einzeln. In Monatseinnahmen stehen dann nur noch die Summen der Artikel und nicht mehr für jede Theke getrennt. Dies soll gewährleisten, dass die Applikation auch bei größeren Datenmengen noch zufriedenstellend schnell arbeitet.

Um jedoch zu jedem Zeitpunkt noch nachträglich detaillierte Analysen zu ermöglichen, werden die Detaildaten aufgehoben.

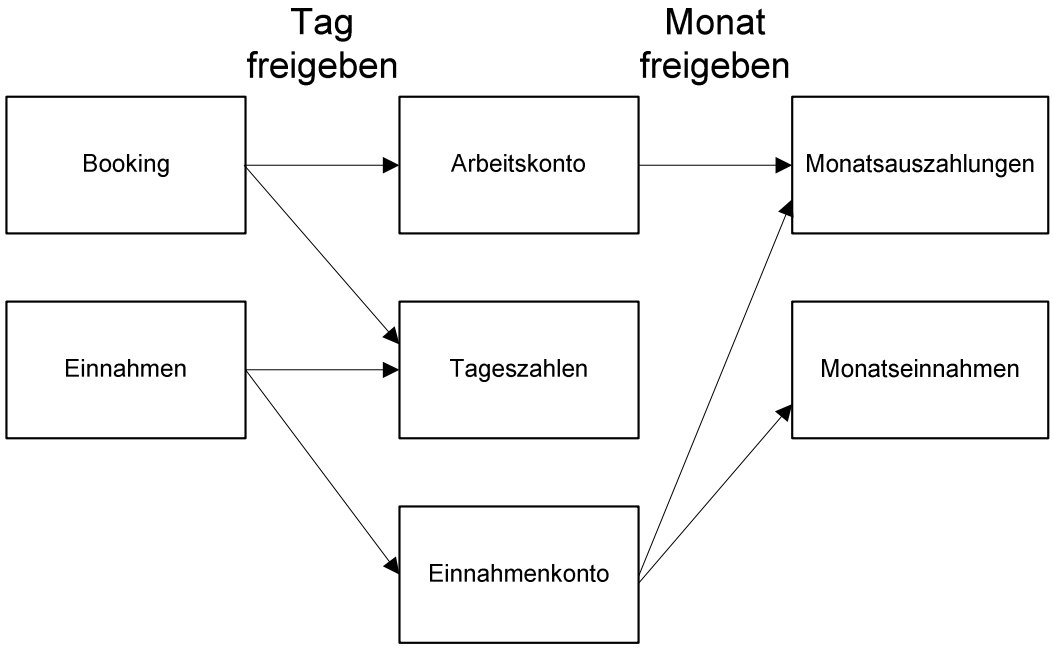


Abbildung 11-6 Tag und Monat freigeben

11.7.1 Tag freigeben

Hinter der Funktion Tag freigeben, verbirgt sich in der Applikation eine Seite, auf der man alle Tage des selektierten Monats angezeigt bekommt und den Tag auswählen kann, der freigegeben werden soll. Dies löst einen Mechanismus aus, der in Abbildung 11-6 zu sehen ist und wie dargestellt mehrere Tabellen betrifft.

Boking, Arbeitskonto und Tageszahlen

Booking enthält relativ abstrakte Buchungen von Mitarbeitern. Diese Buchungen sind in sofern abstrakt, als in ihnen nur gespeichert ist, wann welcher Mitarbeiter wo arbeitet. In dieser Tabelle ist nicht die Information enthalten, wieviel Lohn der Mitarbeiter für den Zeitraum erhält oder wieviel Steuern er für seinen Lohn für den Zeitraum bezahlen muss. Diese Informationen werden erst geschrieben, wenn

der Tag freigegeben wird. Dann werden diese Werte für jeden Mitarbeiter, der an diesem Tag gearbeitet hat, in die Tabelle Arbeitskonto geschrieben. An dieser Stelle werden die Daten konkret unter Einbeziehung des aktuellen Stundenlohnes berechnet und komplett in die Datenbank geschrieben, damit jederzeit Änderungen des Stundenlohnes möglich sind, ohne dass alte Daten inkonsistent werden. Im gleichen Schritt werden die Summen dieser Werte gesammelt in die Tabelle Tageszahlen geschrieben, die später die Auswertung des Abends, also die Abendabrechnung repräsentiert.

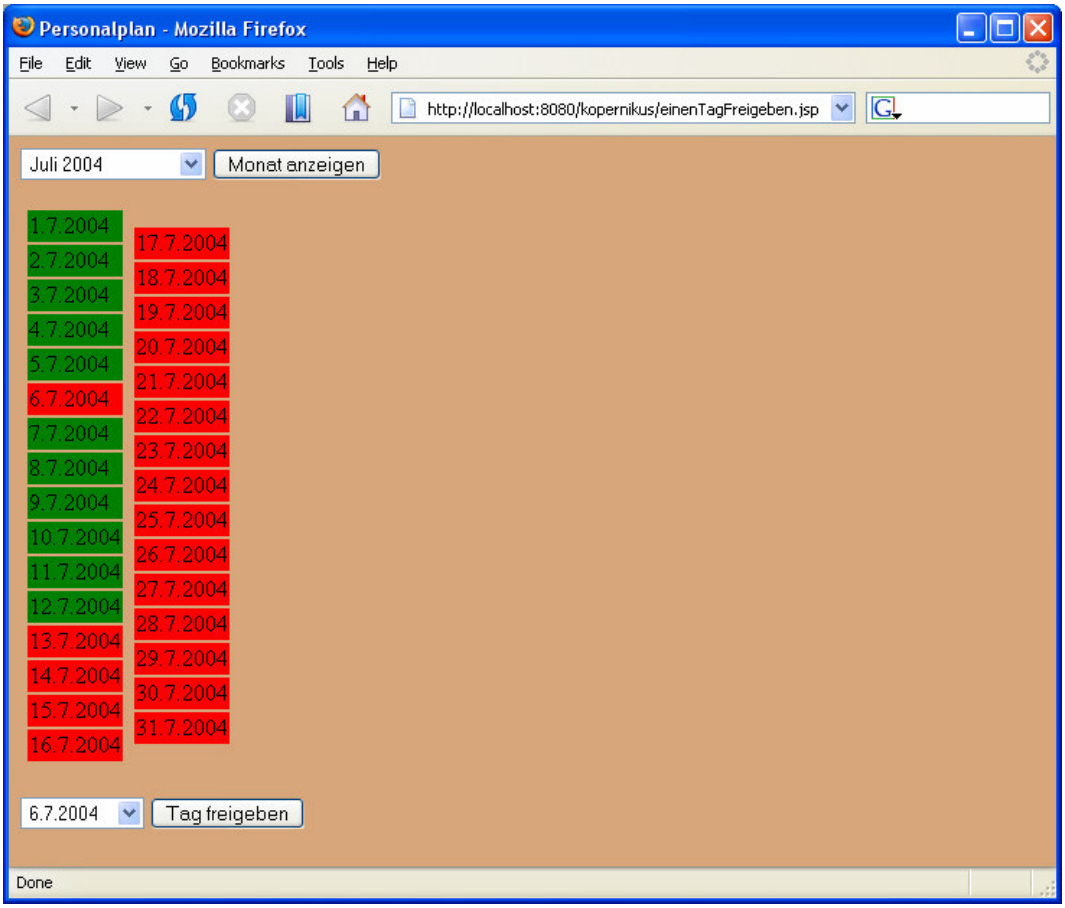


Abbildung 11-7 Einen Tag freigeben

MitarbeiterID	BereichsID	VonDatum	VonZeit	KW	BisDatum	BisZeit
5	1	14.12.2004	20:00	52/2004	15.12.2004	3:00
17	3	14.12.2004	20:00	52/2004	15.12.2004	2:00

Tabelle 6 Table Booking

MitarbeiterID	Datum	Was	Wieviel	Wer
2	29.5.2004	Bruttolohn	40.0	Arbeitnehmer
2	29.5.2004	Rente	2.0	Arbeitnehmer
2	29.5.2004	Sozialabgaben	4.0	Arbeitnehmer
2	29.5.2004	Sozielabgaben	4.0	Arbeitgeber
2	29.5.2004	Nettolohn	34.0	Arbeitnehmer

Tabelle 7 Table Arbeitskonto

Art	Datum	Was	Wer	Wieviel
Steuer	29.5.2004	Sozielabgaben	Arbeitgeber	8.0
Steuer	29.5.2004	Rente	Arbeitgeber	2.0
Ausgabe	29.5.2004	Bruttolöhne	Arbeitnehmer	110.0

Tabelle 8 Table Tageszahlen

Einnahmen, Einnahmenkonto und Tageszahlen

Die Tabelle Einahmen enthält detailierte Infromationen zu jeder Kasse. So sind beispielsweise die verbuchten Mengen jedes Artikels zu jeder Kasse und jedem Datum einzeln gespeichert. Wenn nun der Tag freigegeben wird, werden die Summen aller gebuchten Artikel kassenweit gebildet und in die Tabelle "Tageszahlen" geschrieben, damit mit diesen Summen die Abendabrechnung vorgenommen werden kann. Damit man später einen schnellen Zugriff auf die konkreten Steuern und Gewinn/Umsätze der Artikel hat, werden diese Werte parallel in die Tabelle "Einnahmenkonto" geschrieben.

Datum	Kasse	Artikel	Menge
16.4.2004	1	14	5
17.4.2004	4	11	3

Tabelle 9 Table Einnahmen

Datum	Kasse	Artikel	Umsatz	Menge	Steuer	Ort	Gewinn
29.4.2004	3	4	2.0	1	0.14	Theke	0.76
29.4.2004	5	7	1.0	1	0.07	Theke	0.43
29.4.2004	1	5	3.0	3	0.21	Bar	1.29

Tabelle 10 Table Einnahmenkonto

Art	Datum	Was	Wer	Wieviel
Einnahme	29.5.2004	Umsatz	Arbeitgeber	32.0
Einnahme	29.5.2004	Einnahmen-Gewinn	Arbeitgeber	15.0

Tabelle 11 Table Tageszahlen

11.7.2 Monat freigeben

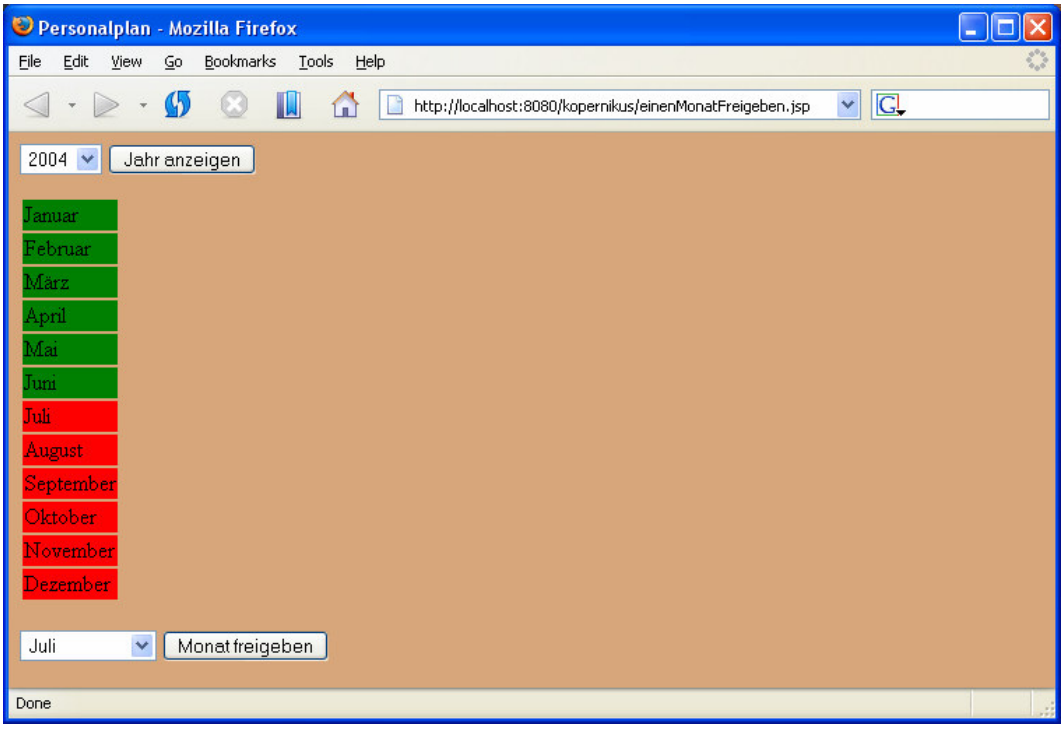


Abbildung 11-8 Einen Monat freigeben

Diese Seite sieht ähnlich aus wie die der Freigabe für die Tage, nur, dass auf dieser Seite keine Tage, sondern Monate auszuwählen sind. Es lassen sich hier nur Monate auswählen, deren Tage schon komplett freigegeben sind. Dies stellt unter anderem sicher, dass die Löhne erst dann ausgezahlt werden, wenn alle Mitarbeiterbuchungen bestätigt worden sind.

Arbeitskonto und Monatsauszahlungen

Wenn ein Monat freigegeben wird, werden zunächst die Brutto-Löhne der einzelnen Arbeitstage eines Mitarbeiters addiert und als Summe zur Auszahlung an den Mitarbeiter in Monatsauszahlungen geschrieben.

Aber nicht nur die Summe der Löhne der Mitarbeiter, sondern auch die Summen der Lohnnebenkosten werden in die Tabelle Monatsauszahlungen übertragen.

Monat	Wer	MitarbeiterID	Was	Wieviel	Gezahlt
4/2004	Arbeitnehmer	2	Lohn	36.0	36.0
4/2004	Arbeitnehmer	12	Lohn	63.21	0.0
4/2004	Finanzamt	0	Rente	6.55	6.55
4/2004	Finanzamt	0	Sozialabgaben	12.58	12.58

Tabelle 12 Table Monatsauszahlungen

Einnahmenkonto, Monatseinnahmen und Monatsauszahlungen

Beim Freigeben werden die Informationen der einzelnen Kassen nach Theken[32] summiert in die Tabelle Monatseinnahmen übertragen. Dies ermöglicht ein schnelles, effizientes Darstellen der Umsätze, Gewinne usw. in der Applikation, da die Werte nicht jedesmal neu berechnet werden müssen, sonder konkret vorliegen.

Datum	Woher	Umsatz	Gewinn
29.4.2004	Theke	4.0	1.72
29.4.2004	Cafebar	5.0	3.04
12.5.2004	Cafebar	10.0	6.08

Tabelle 13 Table Monatseinnahmen

Die entsprechenden zu entrichtenden Steuern, für die im Einnahmenkonto verbuchten Artikel, werden im gleichen Schritt in die Tabelle Monatsauszahlungen nacht Steuerart summiert übertragen.

Monat	Wer	MitarbeiterID	Was	Wieviel	Gezahlt
2/2004	Finanzamt	0	Mehrwertsteuer	12.15	12.15
3/2004	Finanzamt	0	Mehrwertsteuer	35.44	35.44
4/2004	Finanzamt	0	Mehrwertsteuer	22.12	0.0

Tabelle 14 Table Monatsauszahlungen

11.7.3 Tag Unfreigabe

Die Unfreigabe des Tages ist das passende Gegenstück zur Freigabe. Sie nimmt all das zurück, was die Freigabe veranlasst hat. Ein Grund hierfür kann beispielsweise sein, dass ein Mitarbeiter aus Versehen gebucht wurde, der an dem betreffenden Abend gar nicht gearbeitet hat. Wenn dies dem verantwortlichen Zapfer[33] am betreffenden Abend nicht auffällt, dann wandern seine Stunden,

[32] eine Theke kann mehrere Kassen haben, aber jede Kasse kann nur an einer Theke stehen.

[33] der Abendverantwortliche

obwohl er nicht gearbeitet hat, in die Abrechnung. Wenn dies am Ende des Monats, oder früher auffällt, kann der Monat unfreigegeben werden, die Daten angepasst werden und der Tag im Anschluß wieder freigegeben werden.

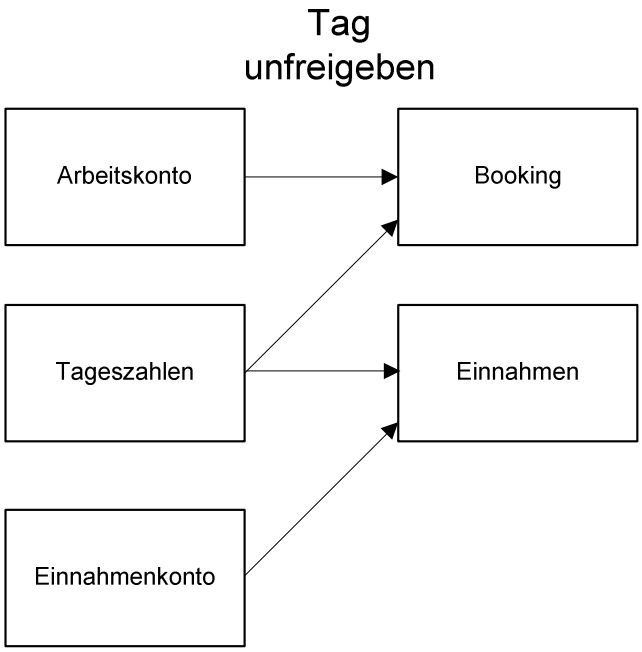


Abbildung 11-9 Einen Tag unfreigeben

Hierfür gibt es keine Entsprechung für den Monat, da nach Freigabe des Monats eventuell schon Gehälter ausgezahlt oder Steuerschulden beglichen wurden. Die Daten in diesem Falle konsistent zu halten ist quasi unmöglich.

12 Modul Kassensystem

Das Kassensystem dient ausschließlich zum Buchen von Artikeln und zeigt dabei lediglich die an diesem Kassensystem zuletzt gebuchten Artikel, Mengen und die Summe der Einnahmen an. Es dient nicht zur Auswertung bzw. Abrechnung. Für diesen gibt es ein in den seperaten Modulen Statistik/Auswertung, beziehungsweise der Abendabrechnung die entsprechnde Funktionalität.

Das komplette Kassensystem ist so entwickelt, dass es den Ansprüchen der Umgebung, in der es arbeitet, Folge leistet. So ist beispielsweise in einer solchen Umgebung meist nur das Aufstellen eines monochromen[34] 12”-14” Monitor möglich. Deshalb arbeitet das Kassensystem bei einer Auflösung von 640 x 480 und ist farblich auf die Farben schwarz und weiß reduziert.

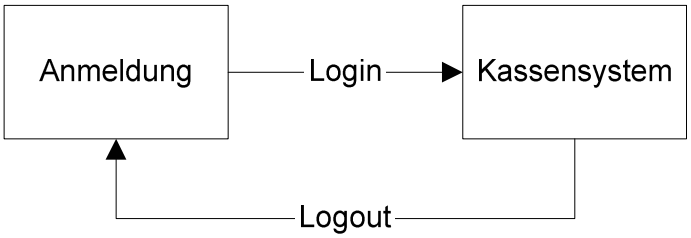


Abbildung 12-1 Ablauf Kassensystem

12.1 Locking

Zunächst gelangt man auf eine Seite, auf der die noch freien Kassen angezeigt werden (siehe Abbildung 12-2). Um die Kasse anzumelden, wählt man aus diesen Kassen eine aus und klickt auf anmelden.

[34] schwarz-weiß Monitor

Wenn nun eine andere Person ebenfalls auf die Anmeldeseite geht, während man selbst sich noch nicht angemeldet hat, bekommt sie die gleichen Kassen angezeigt. Wenn nun der unwahrscheinliche Fall eintritt, dass der andere genau diese Kasse auch auswählt, würden beide Personen über eine Kasse buchen. Das Kassensystem würde in diesem Fall zwar fehlerlos arbeiten, aber es handelt sich dabei um eine sicherlich unerwünschte Situation.

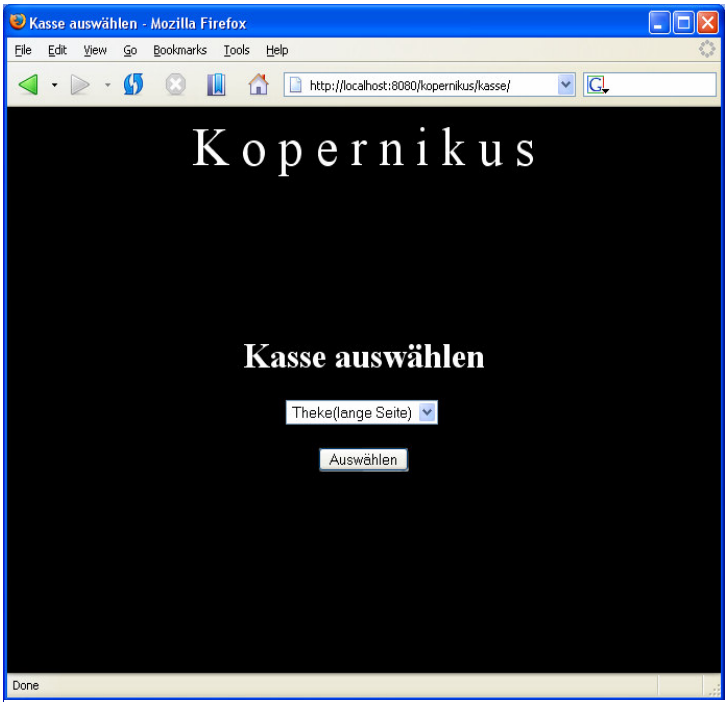


Abbildung 12-2 Anmeldung Kassensystem

Um zu verhindern, dass beide dieselbe Kasse verwenden, wird ein Locking-Mechanismus verwendet. Wenn man die Kasse ausgewählt und angemeldet hat, wird auf der nächsten Seite in der Tabelle, in der der Kassenstatus steht, für die betreffende Kasse ein belegt eingetragen. Das Vorhaben einer zweiten Person sich fälschlicherweise an der Kasse anzumelden scheitert, da das Kassenapplet vor der Ausführung überprüft, ob die Kasse noch frei ist.

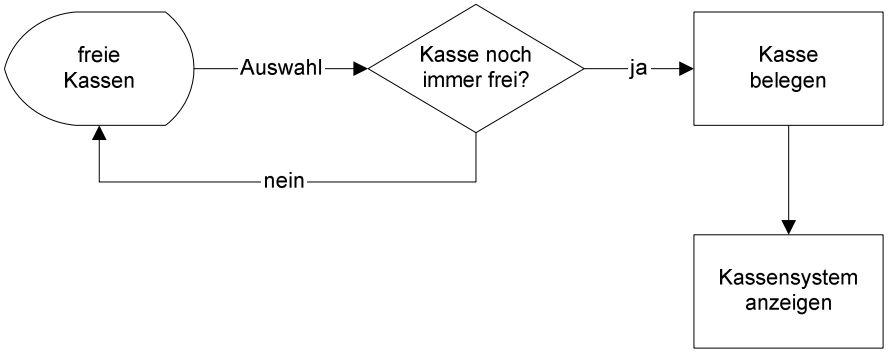


Abbildung 12-3 Locking-Mechanismus

In der Praxis sollte dieser Fehler jedoch nie auftreten, da ja dort ohnehin jede physikalisch existierende Kasse ihre eigene Bezeichnung hat. Da diese

Problematik also mehr theoretischer Natur ist, wurde auf eine noch zeitkritischere Absicherung verzichtet.

Ein wichtiges Implementierungsdetail wäre vielleicht noch, dass das Freigeben der Kasse nicht ohne eine Bedingung in der destroy-Methode erfolgen sollte. Diese Methode wird ausgeführt, sobald ein Applet beendet wird. Was im Fall einer normalen Belegung der Kasse funktioniert, verursacht einen Fehler, wenn das Applet ausgeführt wird und keine Kasse belegt wird, da sie schon belegt ist. Die Kasse würde ohne Rücksicht auf den anderen, der die Kasse gerade nutzt, nach dem Beenden des eigenen Applets freigegeben werden. Dies muss dadurch verhindert werden, dass die Kasse nur freigegeben wird, wenn der aktuelle Benutzer diese auch bekommen hat.

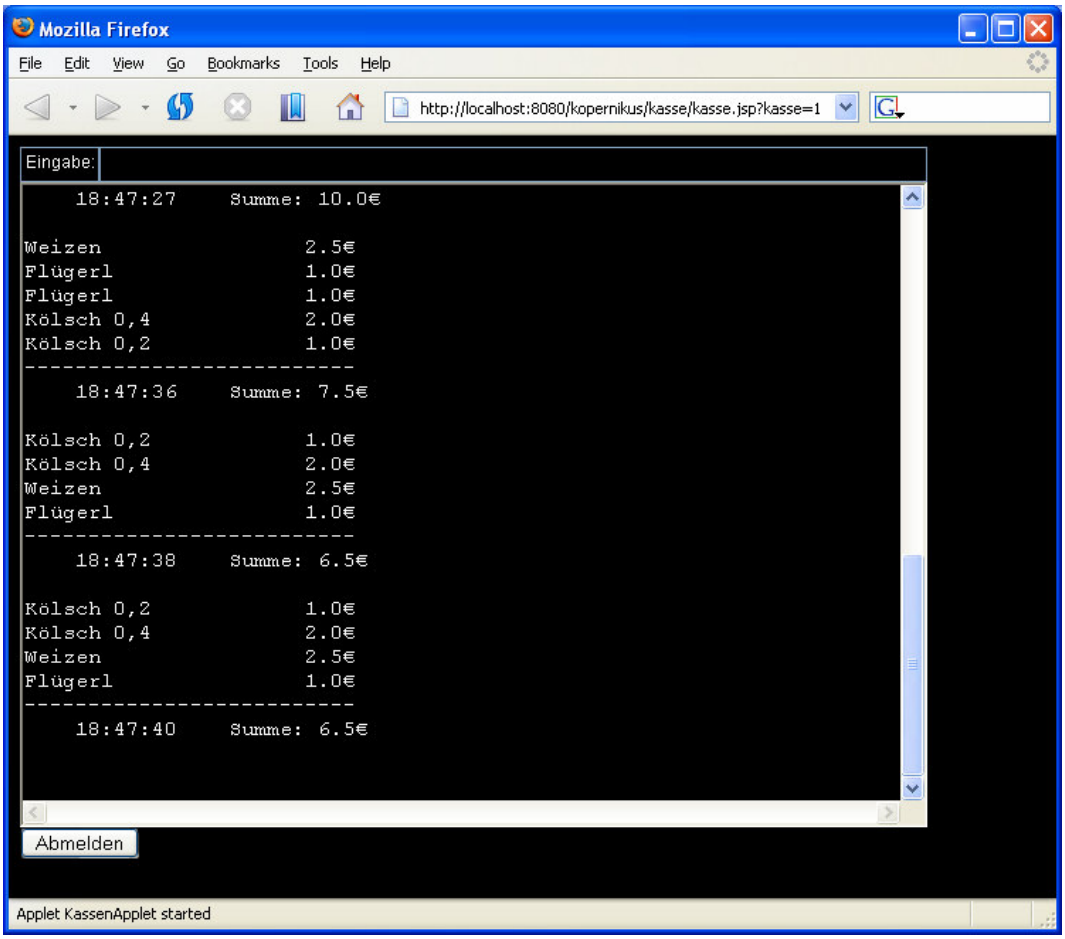


Abbildung 12-4 Das Kassensystem

Sobald man nun also eine noch freie Kasse ausgewählt hat, gelangt man auf die nächste Seite, welche das eigentliche Kassensystem darstellt, und mit der man die verkauften Artikel verbuchen und zusammenrechnen kann.

Das Verbuchen eines Artikels erfolgt über den Druck, eines für jeden Artikel beim Anlegen (siehe Kapitel 14.1.1) definierten Tastenkürzels. Dieses Tastenkürzel, ist als ASCII[35]-Code-Zeichen gespeichert. Zum Tragen kommt der Umstand mit den ASCII-Zeichen insbesondere dann, wenn man eine Spezialtastatur verwendet, die keinem QWERTZ-Layout[36] entspricht.

Wird nun das entsprechende Tastenkürzel für einen Artikel gedrückt, erscheint dieser zunächst mit seinem Preis im Ausgabefenster. Dort wird so lange nach jedem Kürzeltasten-Druck ein Artikel unten angehangen, bis der Nutzer Enter drückt. Das drücken der Enter Taste veranlasst, dass die Summe der gebuchten Artikel gebildet und ausgegeben wird. Das Kassensystem fängt nun wieder bei null an zu zählen.

Solange das Kassensystem nicht zwischenzeitlich geschlossen wurde, lassen sich alle am Abend vorgenommenen Buchungen nachträglich nachlesen, da alle neuen Zeilen angehangen werden und sich zu den alten hoch scrollen lässt.

Es ist selbstverständlich nach der Anmeldung der Kasse nur möglich, an dieser Kasse Artikel zu buchen, welche der Theke an der sich die Kasse befindet, beim Anlegen des Artikels zugewiesen wurde.

---

[35] [InfTa] ISO-7-bit-Code/ASCII (American Standard Code for Information Interchange). Durch die Internationale Referenz-Version(ISO/IEC 646) genormt. Mit dieser ust due US-Referenz(ASCII) identisch. Für die deutsche Referenz gilt DIN 66 003- Jedes Zeichen wird mit 7 bit dargestellt. Die Bedeutung der Bitkombinationen entnimmt man der genormten ASCII-Tabelle.

[36] Das QWERTZ Layout ist ein genormtes Layout für eine deutsche Standard Tastatur. Die Buchstaben stehen für die ersten Buchstaben der ersten Buchstabenreihe von oben. Dafür gibt es auch im englischen ein entsprechend genormtes Layout. Hier stehen in der ersten Buchstabenreihe die Buchstaben QWERTY. Somit ist die englische Standart Tastatur eine QWERTY-Tastatur.

Die Artikel, die gebucht wurden, werden alle einzeln, das heißt nach Artikeln aufsummiert in der Datenbank gespeichert. Für jeden Artikel, zum Beispiel können dies Kölsch 0,2l oder Weizen sein, wird für jeden Tag eine neue Zeile angelegt. Dies ermöglicht spätere Statistiken, bzw. eine Inventur, da man die Menge an einzelnen verkauften Waren pro Tag erfasst hat.

Falls irgendwann mal ein Artikel falsch gebucht wurde, gibt es jederzeit die Möglichkeit, einen Artikel heraus zu buchen. Dies ist auch noch möglich, nachdem bereits die Summe gebildet wurde. Es wird eingeleitet durch ein Drücken der Minus, oder Bindestrich Taste und anschließendes Drücken des Kürzels des zu löschenden Buchstaben.

13 Modul Deejaying

13.1 Playlist

Die Playlist ist eine Liste, in der die am Abend gespielten Interpreten und Titel in der gespielten Reihenfolge aufgelistet sind. Mit Hilfe dieser Liste können Gäste zum Beispiel zu einem Lied den Titel oder Interpreten ermitteln. So ist es häufig der Fall, dass Gäste am Ende des Abends oder mittendrin fragen, wie ein bestimmter Titel hiess. Diese Anfragen lassen sich häufig vermeiden, wenn die Gäste Zugriff auf eine Playliste haben. Dies ist auch das Verfahren, was alle großen Radiosender durchführen um die Fülle der Anfragen der Hörer einzuschränken.

13.1.1 Hinzufügen

Hier kann der Deejay herausfinden, auf welcher CD im Archiv sich ein bestimmter Titel befindet. Er kann hierzu entweder nach dem Titel oder nach dem Interpreten suchen.

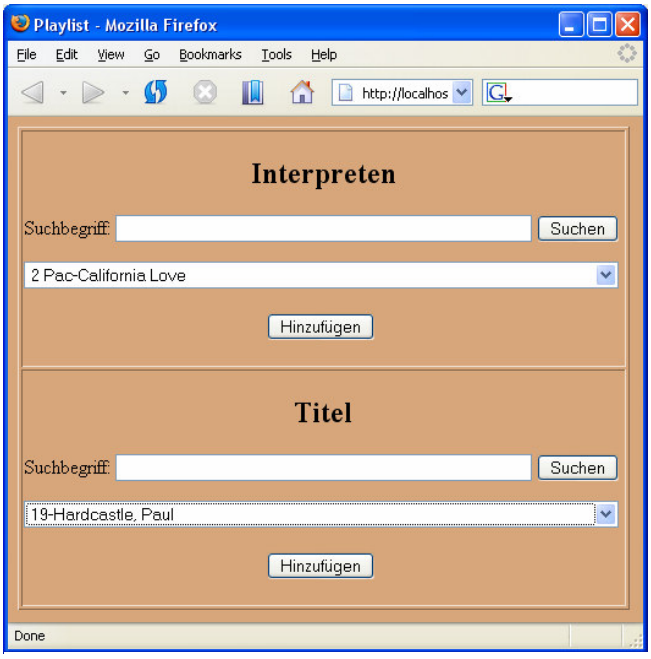


Abbildung 13-1 Titel einer Playlist hinzufügen

Wenn er sich dann entschlossen hat, einen Titel zu spielen, wird dieser der Playlist hinzugefügt. Hierfür wird die ID des Titels mit einem Zeitstempel versehen in die Datenbank gespeichert. Der Zeitstempel wird in Millisekunden gespeichert und lässt sich als Schlüssel für eine Sortierung der gespielten Reihenfolge verwenden. Das eigentliche Datum, an dem das Stück gespielt wurde, wird, damit die Suchfunktion effizienter arbeitet, getrennt davon geschrieben.

Hierzu wird der aktuelle Intraday[37] per heuteDatum.getIntraDay() ermittelt und in die Datenbank geschrieben.

Intraday	Reihenfolge	Interpret	Titel	CD
8.6.2004	1086699933328	Anouk	Nobodys wife	A-2
8.6.2004	1086700343578	Max	Can't wait until tonight	M-1
8.6.2004	1086700516687	Ace of Base	All that she wants	A-1

Tabelle 15 Table Playlist

13.1.2 Anzeigen

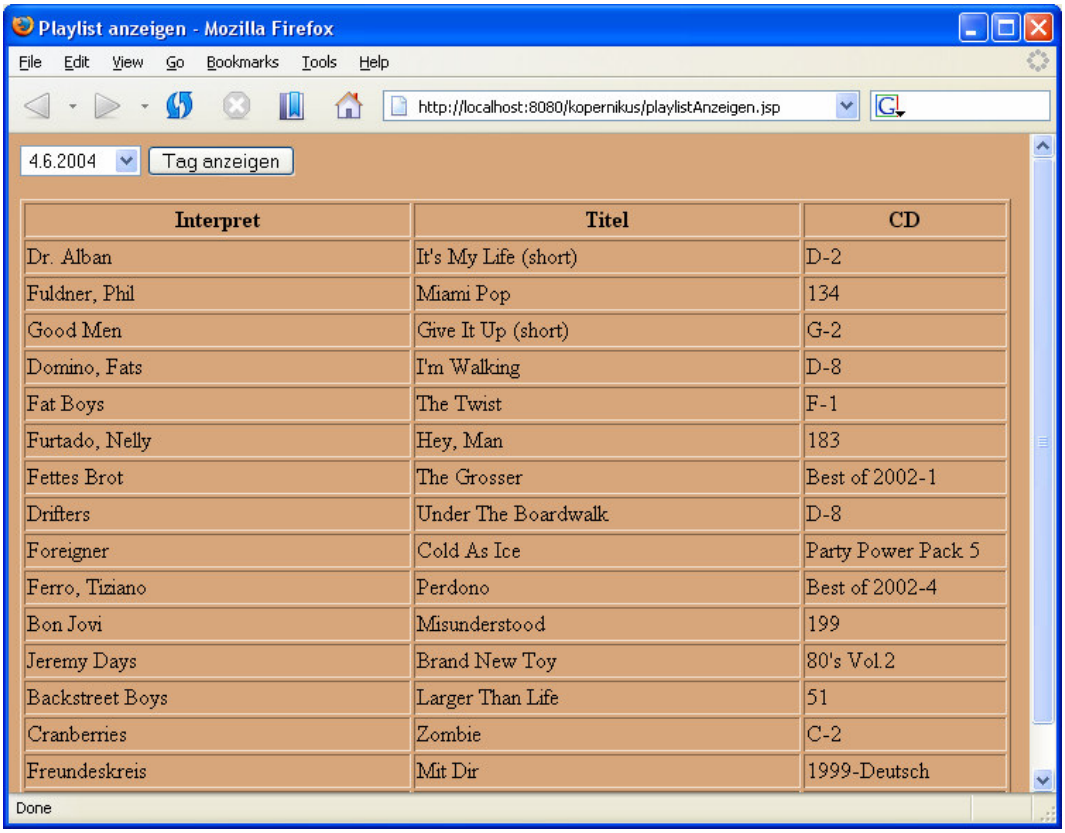


Abbildung 13-2 Playlist anzeigen

Mit dieser Funktion können Gäste oder Mitarbeiter bereits am laufenden Abend oder aber auch später noch einsehen, welche Titel gespielt wurden. Hierzu werden

[37] ein Intraday ist ein Tag, der sich nicht an den allgemein üblichen Tag/Nacht Rhythmus hält, sondern dieser Tag beginnt und endet mit den kontextbezogenen Gegebenheiten. Ein TV-Intraday beginnt und endet beispielsweise um 3:00. Für einen Clubbetrieb ist die Tag/Nacht-Grenze üblicherweise um 8:00 Uhr herum anzusiedeln. Das ist auch die Grenze, welche diese Applikation für einen Intraday verwendet.

die Datensätze, die unter dem ausgewählten Intraday gepeichert werden aus der Datenbank gelesen. Die Reihenfolge wird dann über den als long gespeicherten Zeitstempel sortiert hergestellt.

13.2 Wunschliste

Im Laufe eines Abends treten eine Menge Gäste an den Deejay heran und möchten sich einen Titel wünschen. Die Wünsche erzeugen einen Zettelwust, den diese Applikation mit der Wunschliste verhindert. Mit der Wunschliste wird eine Funktion geschaffen, welche die Wünsche der Gäste zentral aufnimmt und verwaltet. In der Praxis trägt der Deejay hier über die Funktion hinzufügen nach jedem Wusch den gewünschten Titel ein. Diese Funktion handhabt nicht nur die Wünsche der Gäste, sondern auch, wie oft jeder Titel von den Gästen gewünscht wurde.

13.2.1 Hinzufügen

Wenn man einen Musikwunsch hinzufügen möchte, gelangt man erst zu einem Formular, in dem man entweder nach Titel oder Interpreten suchen kann. Die Suchergebnisse, die zum Suchbegriff passen, sind in einem Dropdownmenu auswählbar.

Sobald dann ein Lied ausgewählt ist und es hinzugefügt wird, wird in der Datenbank überprüft, ob dieser Titel bereits gewünscht ist. Ist dies der Fall wird die Zahl der Wünsche für diesen Titel incrementiert, falls nicht wird ein neuer Datensatz angelegt.

13.2.2 Anzeigen

Hier können die Gäste, beziehungsweise der Deejay einsehen, welche Titel bereits gewünscht wurden und wie oft.

13.3 Archiv

Im CD-Archiv ist zu allen verfügbaren Titeln und Interpreten gespeichert, auf welchem Datenträger sie sich befinden. Dies ermöglicht dem Deejay eine schnelle Suche, damit er sich ganz auf seine eigentliche Arbeit konzentrieren kann. Eine

ausschließliche Suche über das Gedächtnis ist, obwohl es natürlich eine Systematik gibt, bei meist um die 1000 Platten und ca 19.000 Titeln quasi unmöglich.

13.3.1 Import

Diese Funktion dient dazu, dass die Deejays Plattensammlungen in das Programm importieren können. Wie in Kapitel 5.5.2 bereits erläutert, müssen die zu importierenden Daten als csv-File vorliegen. Dieses File muss im Tomcat-Root Directory abgelegt werden, damit es importiert werden kann.

Wenn man nun die import-Funktion des Programms aufruft, wird man aufgefordert, den Dateinamen einzugeben. Die angegebebene Datei wird nun mit Hilfe von FileDataHandler (siehe 7.2) geöffnet und alle Zeilen ausgelesen.

ID	Interpret	Titel	CD
195	Baha Men	Who let the dogs out	B-2
196	Baloon	Monstersound	B-2
197	Bangles	Manic Monday	B-2
198	Bangles	Walk like an Egyptian	B-2

Tabelle 16 Table CDArchiv

13.3.2 Export

Mit Hilfe dieser Funktion kann das momentane CD-Archiv in ein externes csv-File geschrieben werden. Dies kann zum Beispiel nötig sein, um das CD-Archiv in eine andere Applikation zu exportieren, oder wenn der Deejay sich das CD-Archiv zu Hause (in Excel) ansehen möchte.

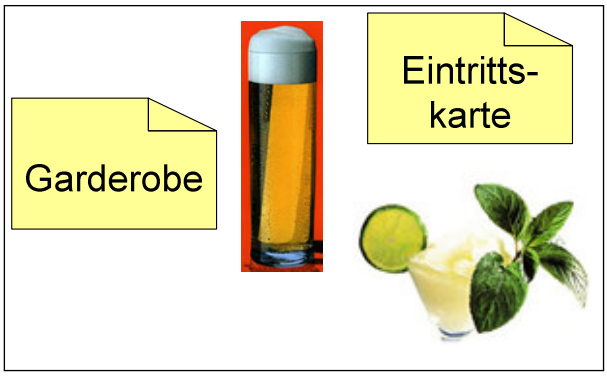
Das CD-Archiv, welches mit dieser Funktion exportiert wurde, landet nach dem Export im gleichen Verzeichnis, wie die import-Daten erwartet werden, also dem Tomcat-Root.

14 Modul Administration

Diese Modul stellt die administrativen Funktionalitäten der Applikation zur Verfügung.

14.1 Artikel

Bei jedem, mit Hilfe der Applikation verkauftem Gut, handelt es sich um einen Artikel. Das bedeutet, dass es sich bei Garderoben- oder Eintrittskarten ebenso um Artikel handelt wie zum Beispiel bei einem Getränk.



ID	Name	Beschreibung	VK Preis	EK Preis	Kürzel	Artikelgruppe	Ort
34	Kölsch 0,2	Kölsch im 0,2 Glas	1.00	0.44	k	mit Alkohol	Theke
35	Kölsch 0,4	Kölsch im 0,4 Glas	2.00	0,88	g	mit Alkohol	Theke
36	Wasser	Wasser im 0,2 Glas	1.00	0.22	w	ohne Alkohol	Theke
7	Eintritt	Eintritt	4.00	0.00	e	diverses	Eingang

Tabelle 17 Table Artikel

14.1.1 Anlegen

Mit diesem Dialog legt man neue Artikel an, die später mit dem Kassensystem gebucht werden können.

Um einen Artikel anzulegen, bestimmt man erst einen Namen und weist diesem Artikel dann eine Theke zu, an der er verkauft werden soll. Jede Theke und somit die an ihr befindlichen Kassensystem verfügen über einen eigenen Pool an Artikeln. Wie im Kapitel 12 über das Kassensystem bereits erwähnt, muss hier dem Artikel dann als nächstes ein Tastenkürzel beigebracht werden. Hierfür

navigiert man in das entsprechende Eingabefeld und drückt auf der Tastatur des Kassensystems die zuzuweisende Taste.

Nachdem man den EK-[38] und den VK-Preis[39] bestimmt hat, kann man noch eine Artikelgruppe angeben, welche momentan noch mit keiner Funktionalität versehen ist. Die Artikelgruppe soll später dazu dienen, Artikel auf einer automatisch generierten Preisliste zu gruppieren.

14.1.2 Anzeigen

Hier bekommt man eine nach Theken sortierte Übersicht, über die buchungsfähigen Artikel angezeigt. Diese Übersicht zeigt neben EK- und VK-Preis jeweils auch die Kürzel und die Beschreibung zum jeweiligen Artikel an.

14.1.3 Löschen

Beim Löschen eines Artikels kommt die gleiche Problematik wie schon beim Löschen eines Mitarbeiters (siehe Kapitel 9.4) zum Tragen. Der Artikel ist per ID in einigen anderen Tabellen indiziert. Ein Löschen des Artikels hätte unweigerlich Inkonsistenzen zur Folge. Deshalb wird wie auch bei den Mitarbeitern zwischen aktiven und passiven Artikeln unterschieden. Ein Artikel wird hier also ebenfalls durch das Löschen nicht aus der Datenbank gelöscht, sondern er wird lediglich auf passiv gesetzt.

14.2 Theke

14.2.1 Anlegen

Eine Theke kann eine oder mehrere Kassen enthalten. Um eine solche Theke anzulegen, welcher später beim Anlegen der Kassen, Kassen zugewiesen werden, braucht es nur einen Namen und eine Beschreibung für diese Theke.

---

[38] steht für Einkaufs-Preis und stellt so den Preis dar, zu dem ein Artikel pro Mengeneinheit eingekauft wird.

[39] Verkaufs-Preis ist das logische Gegenstück zum Einkaufspreis und kennzeichnet somit den Preis, zu dem der Artikel je Mengeneinheit verkauft wird.

14.2.2 Anzeigen

In dieser Übersicht werden alle verfügbaren und zuweisbaren Theken nach Name sortiert aufgelistet.

14.2.3 Löschen

Hier wird zunächst aus einer Liste der verfügbaren Theken die Theke ausgewählt, welche man löschen, beziehungsweise deaktivieren möchte. Ist dies erfolgt, wird die entsprechende Theke nach einer Bestätigung durch den Benutzer in der Datenbank als unbelegbar, beziehungsweise entfernt markiert und der Eintrag erscheint in dem entsprechenden Dialogen “Kasse auswählen”, beziehungsweise “Kasse anzeigen” nicht mehr.

14.3 Kasse

14.3.1 Anlegen

Jede Kasse, welche im Club verwendet wird, inklusive der Kassen an der Garderobe und dem Eingang müssen zunächst mit dieser Funktion angelegt werden. Mit dieser Funktion ist es auch nachträglich möglich, Kassen anzulegen, wenn zum Beispiel der Club umgebaut wurde, oder an einer Theke eine zusätzliche Kasse aufgebaut wurde.

14.3.2 Anzeigen

Mit dieser Funktion kann man sich alle im Moment im Club aufgestellten Kassen und ihren Status anzeigen lassen. Hier kann man zum Beispiel sehen, was eine Kasse bisher am Abend umgesetzt hat. Hierfür startet die jsp-Seite eine Anfrage an die Kassen-Klasse, welche ihrerseits ein einfaches SQL-Statement der Form “SELECT * FROM Einnahmen WHERE Kasse=’KassenID’ AND Datum=’heute’;” generiert und an die SQLDataHandler-Klasse zur Ausführung übergibt. Die einzelnen ResultSets werden dann zeilenweise aufgeschlüsselt von der Seite angezeigt. Zusätzlich hierzu lässt sich auf dieser Seite natürlich überblicken, welche Kassen momentan eingeloggt sind und welche Kassen zur Zeit nicht verwendet werden.

14.3.3 Löschen

Hiermit lassen sich Kassen, wenn sie nicht mehr benötigt werden, sei es, dass sie abgebaut wurden, oder nur kurzzeitig verwendet wurden, aus der Datenbank löschen, beziehungsweise deaktivieren. Hiernach ist es nicht mehr möglich, diese Kasse auf der Login-Seite auszuwählen und somit auch nicht, sich an ihr anzumelden.

14.4 Vertrag

14.4.1 Anlegen

Jeder Mitarbeiter, der eingestellt wird, bekommt einen Vertrag zugewiesen. Dieser Vertrag beschreibt zum Beispiel, wie viele Soll-Stunden ein Mitarbeiter hat, oder wie er versteuert wird. Damit einem Mitarbeiter nun ein solcher Vertrag zugewiesen werden kann, muss er hier zunächst angelegt und ihm verschiedene Besteuerungen, sowohl Arbeitgeber-, als auch Arbeitnehmeranteile, zugewiesen werden. Von da an steht diese neue Vertragsart jedem bestehenden und jedem neuen Mitarbeiter zur Wahl.

14.4.2 Anzeigen

Hier werden alle für Mitarbeiter verwendbaren und nicht verwendbaren, das heißt die stillgelegten, Verträge angezeigt. Dass die nicht mehr buchbaren Verträge angezeigt werden, hat die Bewandnis, dass man so einsehen kann, was es für Vertragsbedingungen gab, welche vielleicht noch aktuell arbeitenden Mitarbeitern zugewiesen sind.

14.4.3 Löschen

Mit dieser Funktion kann man Verträge löschen, beziehungsweise stillegen, falls diese neuen Mitarbeitern nicht mehr zur Wahl stehen sollen. Alte Mitarbeiter behalten gesetzeskonform ihre alten Verträge weiterhin, lediglich das Zuweisen zu neuen Mitarbeitern wird unterbunden.

14.4.4 Steuer anlegen

Bevor ein neuer Vertrag angelegt werden kann, müssen erst die Steuern angelegt werden, mit denen das Gehalt versteuert wird. Dies können sowohl Arbeitnehmer-, als auch Arbeitgebersteuern sein. Zum Anlegen einer Besteuerung wählt man zunächst einen Namen, dann wie hoch die Steuer sein soll (in Prozent) und schließlich weißt man noch zu, ob diese Steuer zu Lasten des Arbeitnehmers oder zu Lasten des Arbeitgebers geht.

14.4.5 Steuer anzeigen

Hier werden alle Besteuerungen angezeigt, aus welchen sich ein Vertrag zusammensetzen kann.

14.4.6 Steuer löschen

Wenn eine Besteuerung nicht mehr benötigt wird, dann kann sie über diesen Dialog entfernt werden. Dies hat zur Folge, dass diese Steuer als deaktiviert markiert wird und in zukünfigen Verträgen nicht mehr verwendet werden kann.

15 Implementierungsdetails

15.1 Package-Hierarchie

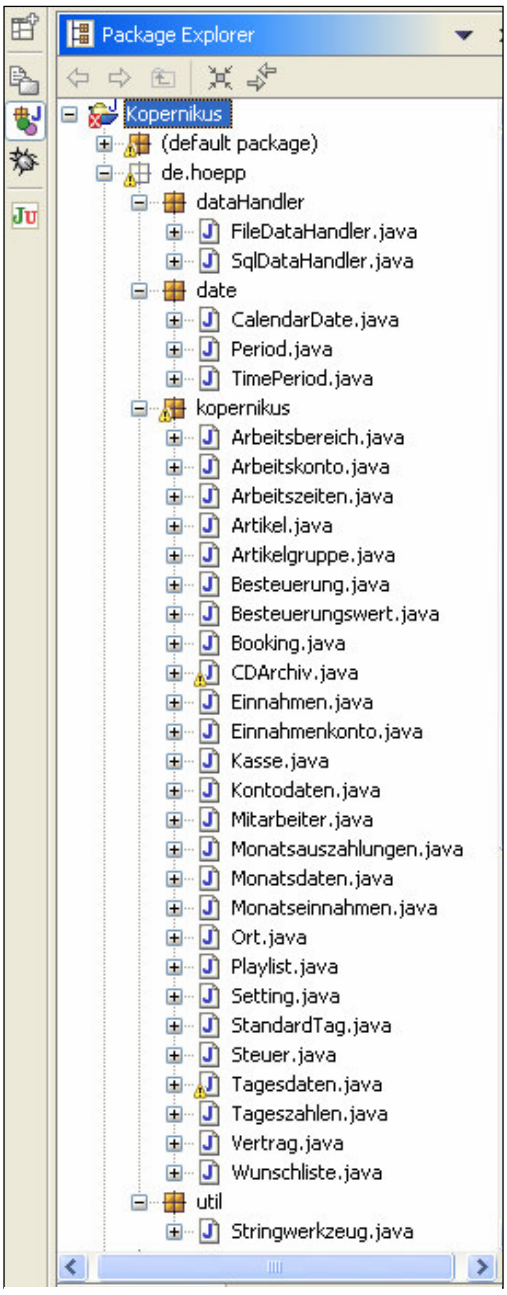
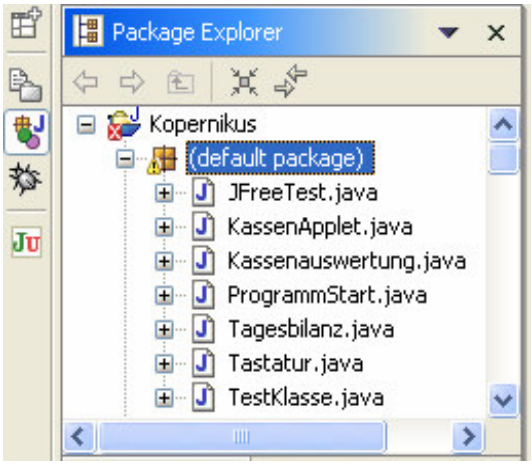


Abbildung 15-1 Die Package Hierarchie



Der Java-Teil der Applikation gliedert sich in fünf Packages unter (siehe Abbildung 15-1).

default Package

Das default Package ist in Abbildung aus Gründen der Übersichtlichkeit zusammengeklappt. Hier befinden sich, neben diversen internen Test-Klassen, die Klassen, welche sich mit der Grafischen Aufbereitung, das heißt mit den jFreeChart (siehe Kapitel 11.6.1)-Darstellungen befassen. Ebenfalls befinden sich hier auch noch alle restlichen Applets, wie zum Beispiel das Kassenapplet, welche die Applikation verwendet.

de.hoepp.date

In diesem Package befinden sich sämtliche

Klassen die sich mit Datum- Arithmetik und der Darstellung befassen.

de.hoepp.dataHandler

Die beiden Klassen, welche sich um die Haltung der Informationen kümmern, also die Klassen SqlDataHandler und FileDataHandler, sind in diesem Package zusammengefasst.

de.hoepp.kopernikus

Hier finden sich sämtliche applikationspezifischen Klassen. Das bedeutet hier finden sich sämtliche Klassen, die keine universelle, für jede Applikation brauchbare, Funktionalität haben. Was allerdings nicht zwangsweise zur Folge hat, dass die Klassen nicht für Applikationen mit einer ähnlichen Funktionalität brauchbar wären. So ist beispielsweise die Klasse Mitarbeiter für jedes Programm funktionell, welches ein Mitarbeitermanagement enthalten soll.

de.hoepp.util

In diesem Package befindet sich in dieser Applikation nur eine Klasse. Die Klasse, welche sich darin befindet, heißt Stringwerkzeug und stellt zusätzliche Funktionalität zum Bearbeiten von Strings bereit, da die Java String Klasse nicht allen Belangen gerecht wird. Zum Beispiel stellt diese Klasse eine Methode bereit, welche es erlaubt, einen String und einen Devider[40] zu übergeben und diese Klasse liefert dann die einzelnen Tokens[41] dazu zurück.

[40] ein Devider ist ein Trennzeichen

[41] als Tokens bezeichnet man die einzelnen Elemente einer Kette.

15.2 de.hoepp.date

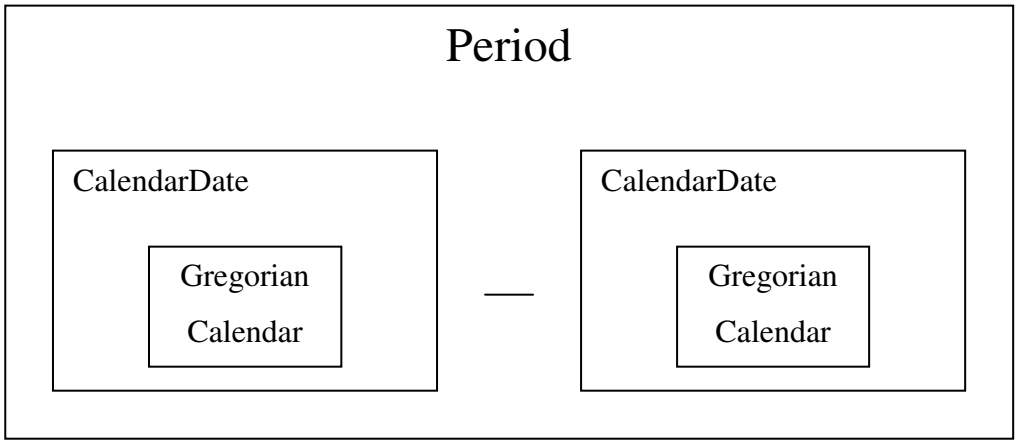


Abbildung 15-2 Veranschaulichung der Klassengeflechts

15.2.1 CalendarDate

Die Klasse GregorianCalendar, die sich in Java mit der nicht trivialen Datumsarithmetik und Darstellung beschäftigt, erfüllt einige Ansprüche an die Funktionalität in der Applikation nicht ausreichend. Deshalb wurde für die Applikation eine Wrapper-Klasse[42], welche GregorianCalendar um die notwendige Funktionalität und Eleganz erweitert, geschrieben.

Der Anlaß und somit die eigentliche Intention die Klasse zu schreiben war, dass der GregorianCalenda Klasse die Möglichkeit fehlt, applikationsweit für ein verbindliches Datumsformat zu sorgen. Diese Notwendigkeit trat auf, als die ersten Datümer in die Datenbank gespiechert werden sollten. Es muss sichergestellt werden, dass ein bestimmtest Format in die Datenbank hinein gespeichert wird und genau aus diesem Format das Eingangsdatum generiert werden kann.

[42] Wrapper-Klassen sind Klassen, welche eine andere umgeben und ihre Methoden und Attribute kapseln, das bedeutet, den Zugriff von außen darauf nicht zu lassen. Die Wrapper-Klassen bedienen sich allerdings der Funktionalität ihrer Klassen, die sie umgeben.

Ein erster einfacher und schnell zu realisierender Lösungsansatz ist sicherlich die TimeInMillis[43] in die Datenbank zu speichern, was auch zunächst die erste Wahl für die Applikation war. Allerdings birgt diese Vorgehensweise eine ganz gravieren Schwäche. Sie birgt Gefahrenpotential, wenn man einen Zeitpunkt speichert und sucht, für welchen die Millisekunden irrelevant sind. Hier kann es bei ungünstiger Implementierung (siehe Abbildung 15-3) dazu kommen, dass die Datumsinformationen im Millisekundenbereich abweichen, auch wenn es sich um den gleichen Zeitraum handelt. Im vorliegenden Beispiel in Abbildung 15-3 sind zum Beispiel die beiden Datümer einDatum und einAnderesDatum ungleich, da sie im Millisekundenbereich voneinander abweichen. Dies ist der Fall, da bei der Anlage über einen leeren Konstruktor das aktuelle Datum inklusive Millisekunden gesetzt wird. Alle Datumsbestandteile werden zwar im Anschluß gesetzt, die Millisekunden aber vergessen. Da beide leeren Konstruktoren zu unterschiedlichen Zeitpunkten[44] im Millisekundenbereich ausgeführt wurden, unterscheiden sich die beiden Daten und somit auch die TimeInMillis. Ein Ausweg ist eine in einer für die Applikation zentral definierte Repräsentation des Datums. Da man nun die Initialisierung mit der Schaffung dieser Klasse selbst in der Hand hat, kann man vollständige Initialisierung verlangen, beziehungsweise ungünstige Zustände oder Zugriffe ganz verhindern.

```
int jahr = Integer.valueOf(getParameter("jahr")).intValue();
int monat = Integer.valueOf(getParameter("monat")).intValue();
int tag = Integer.valueOf(getParameter("tag")).intValue();
int stunden = Integer.valueOf(getParameter("stunden")).intValue();
int minuten = Integer.valueOf(getParameter("minuten")).intValue();
GregorianCalendar einDatum = new GregorianCalendar();
einDatum.set(Calendar.YEAR,jahr);
einDatum.set(Calendar.MONTH,jahr);
einDatum.set(Calendar.DAY_OF_MONTH,tag);
einDatum.set(Calendar.HOUR_OF_DAY,stunden);
einDatum.set(Calendar.MINUTE,minuten);
GregorianCalendar einAnderesDatum = new GregorianCalendar();
einAnderesDatum.set(Calendar.YEAR,jahr);
```

[43] Die Klasse GregorianCalendar hat ein Attribut TimeInMillis, bei dem es sich um einen long handelt, welcher die Millisekunden die seit dem 1.1.1970 vergangen sind beinhaltet und so das aktuelle Datum mit Uhrzeit repräsentiert.

[44] die Wahrscheinlichkeit für die Anlage zur gleichen Millisekunde beträgt 1:1000.

```
einAnderesDatum.set(Calendar.MONTH,jahr);
einAnderesDatum.set(Calendar.DAY_OF_MONTH,tag);
einAnderesDatum.set(Calendar.HOUR_OF_DAY,stunden);
einAnderesDatum.set(Calendar.MINUTE,minuten);
```

Abbildung 15-3 ungünstiges Anlegen zweier identischer Daten

Die Problematik der Abweichungen bei gleichen Daten im Millisekundenbereich lassen sich jedoch auch mit GregorianCalendar-Bordmitteln durch eine anständige Implementierung umschiffen. So hat die Klasse beispielsweise einen Konstuktor, welcher das Anlegen eines Datums ohne Millisekunden vorsieht. Die Milliskunden werden hier automatisch auf null gesetzt. Wenn man nun das Setzen des Datums mit der set-Methode auf ein Objekt, welches mit dem leeren Konstruktor angelegt wurde, verhindert, bewegt man sich im grünen Bereich.

Was allerdings mit GregorianCalendar nicht einfach und applikationsweit zu lösen ist, ist eine einheitliche Repräsentation des Datums per String. Wenn man ein GregorianCalendar Datum als String ausgeben möchte, muss man sich diesen String, wie in Abbildung 15-4 zu sehen, selbst zusammenbauen. Abgesehen davon, dass die Ausgabe des Monats[45] nicht wie erwartet arbeitet, birgt diese Vorgehensweise das Problem, dass nicht applikationsweit sichergestellt ist, dass das Datum immer im gleichen Format ausgegeben und in die Datenbank gespeichert wird. Des Weiteren sieht die Klasse GregorianCalendar keine Möglichkeit vor, dass ein Datum aus einem Datumsstring generiert wird. Das bedeutet, dass man den String bei jedem Lesen eines Datums aus der Datenbank, oder aus der Übergabe aus einer Seite immer per Hand auseinander nehmen muss.

```
System.out.println(einDatum.get(Calendar.DAY_OF_MONTH) + "." +
                   (einDatum.get(Calendar.MONTH)+1)      + "." +
                    einDatum.get(Calendar.YEAR))         + " " +
                    einDatum.get(Calendar.HOUR_OF_DAY)   + ":"+
                    einDatum.get(Calendar.MINUTE)        + ":"+
                    einDatum.get(Calendar.SECOND)
);
```

Abbildung 15-4 Zusammenbauen einer Datum String Repräsentation

[45] die Ausgabe von einCalendar.get(Calendar.Month) gibt den Monat minus eins zurück.

Diese Problematik löst die Klasse CalendarDate dadurch, dass sie Methoden zur einheitlichen Ausgabe eines Datums zur Verfügung stellt und über Konstruktoren verfügt, mit deren Hilfe man aus einem String ein Datum erstellen kann. Natürlich muss auch hier darauf geachtet werden, dass das Datum im richtigen Format übergeben wird. Dies ist allerdings auch schon dadurch sichergestellt, wenn ein Datum aus einem String angelegt wird, welcher vorher per CalendarDate.getDateString() erzeugt und wieder gelesen wurde.

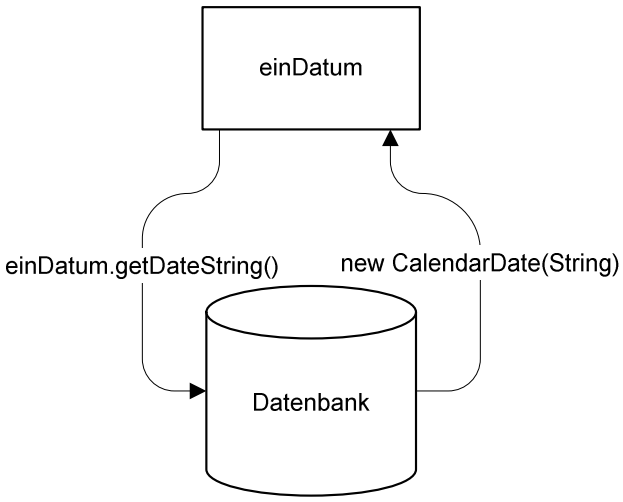


Abbildung 15-5 Datum in Datenbank schreiben und daraus lesen

Wenn man also ein Datum mit der Methode getDateString() in die Datenbank schreibt, lässt sich damit per Konstruktor wieder ein gültiges Datum erzeugen.

Neben diesen genannten Funktionalitäten, verfügt die Klasse CalendarDate noch über eine ganze Reihe anderer sehr nützlicher Funktionen zum Thema Datum, welche der API der Klasse entnommen werden können.

15.2.2 Period

Um die Buchungs- und Abrechnungsfunktion im Mitarbeitermanagement-Modul implementieren zu können, braucht man eine Möglichkeit Zeiträume zu verwalten. Zu diesem Zweck wurde die Klasse Period geschaffen. Sie hat, als ihre beiden Kernattribute[46] zwei Objekte vom Typ CalendarDate, welche einen Anfangs- und einen Endzeitpunkt markieren.

[46] Attribute sind die Variablen/Konstanten einer Klasse

Zunächst einmal muss eine Zeitraumklasse die Möglichkeit bieten, die Anfangs- und Endpunkte nachträglich anpassen und in einem gängigen Format ausgeben zu können. Dies wird dadurch sichergestellt, dass es sich bei den Zeitpunkten um Objekte vom Typ CalendarDate (siehe 15.2.1) handelt, welche all diese Funktionalität bereitstellen.

Als eine der wichtigsten Funktion bei einer solchen Klasse gilt zweifellos die Funktion die Dauer eines Zeitraumes zwischen zwei Zeitpunkten zu bestimmen. Da die Zeitpunkte millisekundengenau erfasst werden, zwingt es sich hier auf, diesen auch in Millisekunden zurückzugeben.

15.2.3 TimePeriod

Die Klasse TimePeriod entspricht in der grundsätzlichen Funktionalität weitestgehend der Klasse Period. Der wesentliche Unterschied der beiden Klassen ist jedoch, dass die Klasse TimePeriod nur Uhrzeit-Zeiträume verwaltet, das heißt Zeiträume ohne Datum.

15.3 Schutz vor falschen Eingaben

Um das Programm vor inkonsistenten Eingaben oder Eingaben im falschen Datenformat zu schützen müssen mehrere Sicherheitsmechanismen etabliert werden. Die Sicherheitsmaßnahmen betreffen sowohl die sendende Seite eines Forms, als auch die Seite, welche die Daten die in ein Form eingegeben wurden, entgegennimmt.

15.3.1 Senden

Überprüfen auf leere Eingaben

Da es gilt die invalide[47] Eingaben schon bei ihrer Entstehung abzufangen, ist es natürlich sinnvoll, die Eingaben schon vor dem Senden zu überprüfen. Hierfür sieht HTML in Formulare die Möglichkeit vor, dass beim Klicken auf den submit-

[47] ungültige

Button[48] ein Java-Script[49] ausgeführt wird. Das Binden eines Java-Script an das dcüken des Knopfes erfolgt folgendermaßen:

```
<input type="submit" value="Speichern"
                onClick="return checkAllesAusgefuellt()">
```

Es wird also veranlasst, dass beim Drücken des "Mitarbeiter anlegen"-Buttons das Script checkAllesAusgefuellt (siehe Abbildung 15-6) ausgeführt wird. Hier wird mit je vier weitgehend identischen Zeilen pro Eingabefeld überprüft, ob ein Feld leer ist und gegebenenfalls eine Hinweis ausgegeben.

```
<script language=JavaScript>
function checkAllesAusgefuellt(){
        if(window.document.Kunde.name.value==""){
                alert("Bitte den Namen eintragen!");
                window.document.Kunde.name.focus();
                return false;
        }
        if(window.document.Kunde.vorname.value==""){
                alert("Bitte den Vornamen eintragen!");
                window.document.Kunde.vorname.focus();
                return false;
        }
        if(window.document.Kunde.Telefon.value==""){
                alert("Bitte eine Telefonnummer eintragen!");
                window.document.Kunde.Telefon.focus();
                return false;
        }
}
</script>
```

Abbildung 15-6 Script zur Validierung der Eingaben

Ob das entsprechende Feld leer ist oder nicht, überprüft das Script zunächst mit if(window.document.FormularName.FeldName.value==““). Falls es tatsächlich leer ist, wird mit alert(„Nachricht“) ein Fenster mit einem Warnhinweis ausgegeben und der Focus[50] per window.document.FormularName.FeldName.focus()

[48] ein Senden Button in einem Formular

[49] eine Scriptsprache, mit der man auch in normalen HTML Seiten über eine gewisse Programmiermöglichkeit von Seiten verfügen kann. Hierdurch lassen sich beispielsweise Validierungen von Werten durchführen

[50] das bedeutet, dass der Cursor in das entsprechende Feld gesetzt ist.

auf das Formularfeld gesetzt. Durch die Rückgabe des false-Wertes wird das Form dazu veranlasst, das Posten[51] zu unterlassen.

Unterbinden der Möglichkeit falscher Eingaben

Eine weitere sehr wirksame Möglichkeit ungültige Eingaben auf der Sender-Seite zu verhindern ist das Verwenden von Auswahllisten. Dadurch, dass man an möglichst vielen Stellen Auswahllisten verwendet, in denen nur gültige Werte zur Auswahl stehen, verhindert man, dass ungültige Werte überhaupt eingegeben werden können.

15.3.2 Empfangen

Da die Werte an jsp-Seiten geschickt werden besteht auf der Empfänger-Seite natürlich die Möglichkeit, Werte umfassend durch die Klassen prüfen zu lassen, welche auch die Objekte aus den Werten erzeugen soll.

Beispielsweise könnte es sein, dass ein Benutzer als Preis für einen Artikel “1,50” eingibt. Dies würde ohne Validierung spätestens bei dem Verkaufen eines solchen Artikels zu einem Fehler führen, da die Beträge als floats[52] verrechnet werden und Floats als Dezimaltrennzeichen einen Punkt verwenden. Also gehört in den Konstruktor der Klasse Artikel natürlich eine Zeile die den eingegebenen String so parst, dass er statt einem Komma einen Punkt als Dezimaltrennzeichen verwendet. Da der Konstruktor allerdings aufgrund anderer Gegebenheiten von vornherein einen float benötigt, muss das parsen entweder per Utility-Methode von der jsp geparst werden, oder bei einfachen Sachverhalten, wie diesem, auch um den Implementierungsaufwand zu begrenzen, in der jsp-Seite geparst werden. Dies erfolgt einfach durch einen regulären Ausdruck, welcher eine String Utility-Methode (siehe Kapitel 7.1.3) übergeben wird:

```
String ekString = (String) request.getParameter("ek");
ekString=ekString.replaceAll(",",".");
float ek=Float.valueOf(ekString).floatValue();
```

[51] als Posten bezeichnet man das Schicken der Werte eines Formulars.

[52] Ein float ist eine Fließkommazahl mit einfacher Genauigkeit, welche in IEEE-754 genormt ist.

16 Zusammenfassung und Ausblick

Die Applikation ist in der voliegenden Version im vorher geplanten Rahmen und darüber hinaus einsetzbar. Das bedeutet, dass die Applikation die Funktionalität bietet, welche nötig ist, einen Club vollständig zu betreiben und zu verwalten. Es sind also alle Funktionen, welche im Pflichtenheft definiert und gefordert wurden auch implementiert worden.

Im Rahmen der Entwicklung der Applikation wurden einige Klassen entwickelt, welche so vielseitig einsetzbar sind, dass sie auch in anderen Applikationen verwendbar sind. So sind beispielsweise die Klassen CalendarDate und TimePeriod hilfreiche Werkzeuge, wenn es darum geht Probleme der Datumsarithmetik zu lösen.

Durch den Aufbau der Applikation in einzelne horizontale Schichten sind Schichten der Applikation komplett für andere Anwendungszwecke nutzbar. Wie weit oben man bei deren Benutzung ansetzen kann, liegt an den Anforderungen, welche an ein hierauf aufbauendes System gestellt werden. So ist es bei vielen Aufgabenstellungen schon möglich, bereits auf der obersten Ebene anzusetzen. Das bedeutet, dass nur die die Informationen repräsentierenden Java Server Pages der obersten Ebene angepasst werden müssen.

Die Einteilung der Applikation in vertikale Schichten, oder besser gesagt in Module, ermöglicht den flexiblen, auf die Anforderungen zugeschnittenen Aufbau der Applikation. Das bedeutet, dass jemand der nur ein Kassensystem braucht, auch nur dieses Modul installieren und einzusetzen braucht. Natürlich lassen sich die Module auch nachträglich noch nachrüsten, ohne dass es zu inkonstenten oder unvollständigen Informationen in der Datenbank kommt.

Entgegen der ursprünglichen Planung wurde das Abrechnungs- und Lohnmodul vollständig entwickelt. Das heißt, dass das entsprechende Modul über die Möglichkeit verfügt, Gehälter zu versteuern und die auszuzahlenden Löhne und

Steuern anzuzeigen. Die Verwaltung von Steuern ist so flexibel ausgelegt, dass auch auf gesetzliche Änderungen einfach reagiert werden kann.

Die Applikation vereint nun alle Funktionen, welche vorher in getrennten Applikationen oder Methoden verfügbar waren in einer Applikation und Datenbank. Es ist ein Produkt geschaffen worden, welches es in der vorliegenden Integration momentan auf dem Markt nicht gibt.

In den nächsten Wochen nach der Diplomarbeit ist es nötig, alle Module weiteren ausgiebigen Tests zu unterziehen, um eine völlige Fehlerfreiheit, welche insbesondere für das Abrechnungsmodul erforderlich ist, zu gewährleisten. Das System darf unter keinen denkbaren Bedingungen Fehler bei der Berechnung der Löhne erzeugen, da dies auch insbesondere unter dem Gesichtspunkt der Versteuerung riskant ist.

Nach den Tests erfolgt der erste Einsatz des Gesamtproduktes (der Deejay-Client wurde ja bereits schon bei der Entwicklung eingesetzt). Nach erfolgreichem ersten Einsatz wird das System dann beständig weiterentwickelt. So ist beispielsweise noch denkbar, die Vernetzung der Applikation mit dem Content für die Internetseite eines Clubs zu etablieren. Ebenso wird eine weitere Integration der Gäste in den Abendablauf angestrebt. So könnte es ein weiteres Modul Kundenkontakt oder Ähnliches geben, welches sich um alle Kunden-, beziehungsweise Gästebelange kümmert. Dieses Modul könnte zum Beispiel eine Funktion Fundsachen oder eine Art Kritikboard beinhalten.

Literaturverzeichnis

Print-Literatur

[InfTa] Uwe Schneider & Dieter Werner
Taschenbuch der Informatik, Fachbuchverlag Leipzig

[Faeskorn] Prof. Dr. Heide Faeskorn-Woyke
Vorlesungsscript Datenbanken und Informationssysteme,
Fachhochschule Köln

[GUIGuide] S. L. Smith & J. N. Mosier
Guidlines for designing User Interface Software
Electronic Systems Division, Mitre Corporation, 1996

[SqlData] Peter Morcinek
Das große SQL Buch, Data Becker

[HtmlData] F. Harms & D. Koch & O.Kürten
Das große (X)HTML & XML Buch, Data Becker

[GotoJava2] Guido Krüger
GoTo Java 2, Addison-Wesley

[Lexi] Das moderne Lexikon in 20 Bänden
Bertelsmann

[SQL-Lernen] M. Ebner
SQL lernen, Addison-Wesley

[jsp1] James Goodwill
Pure JSP, Sams

[tomcat] Amit Bakore
Professional Apache Tomcat, John Wiley & Sons Inc

Online-Literatur

[Injection] SQL-Injection - Angriff und Abwehr
http://www.heise.de/security/artikel/43175
(Juli 2004)

[Lgpl] The GNU Lesser General Public Licence
http://www.object-refinery.com/lgpl.html
(Juli 2004)

[ChartDoku] JFreeChart Developer Guide
http://www.object-refinery.com/jfreechart/guide.html
(Juli 2004)

[SelfHtml] Self-HTML
http://de.selfhtml.org/
(Juli 2004)

[Wiki] Wikipedia Lexikon
http://de.wikipedia.org
(Mai 2004)

[JavaApi] Java 2 Platform Standard Edition 1.4.2 Api Specification
http://java.sun.com/j2se/1.4.2/docs/api/index.html
(Juli 2004)

[W3C] W3Schools Online Web Tutorials
http://www.w3schools.com
(Juli 2004)

Anhang A Installation

A-1 MySql

In der aktuellen Version ist die Installation von MySQL denkbar einfach. Um MySql installieren zu können muss man sich lediglich die aktuelle Version aus dem Internet von der Seite http://www.mysql.org dowloaden und entpacken. Ist das Packet entpackt findet sich in dem angelegten Verzeichnis eine Setup-Datei, welche ausgeführt wird und alles wichtige von selbst erledigt. Die Benutzer der Datenbank können im Anschluss nach der Installation des MySQLAdministrator mit diesem angelegt werden.

A-2 MySql-Connector

Um den MySQL-Connector auf dem Server zu installieren, muss man diesen zunächst ebenfalls unter http://www.mysql.org downloaden und ihn dann in ein Verzeichnis entpacken. In diesem Verzeichnis findet sich im Anschluß eine Datei mit dem Namen mysql-connector-java-3.0.jar. Diese Datei muss nun in das Verzeichnis [JavaHome]\lib\ext kopiert werden.

A-3 MySqlAdministrator

Nachdem man den MySQLAdministrator unter http://www.mysql.org herunter geladen hat, führt man das setup aus und wählt die Standardinstallation. Nach der Installation muss lediglich eine Verbindung zur Datenbank hergestellt werden. Dazu braucht man nur den Benutzernamen, den Namen der Datenbank und das Passwort für die Datenbank.

A-4 MySqlControlCenter

Das MySQLControlCenter muss zunächst wie alle anderen MySQL-Tools von http://www.mysql.org herunter geladen werden. Im Anschluss daran wird das Setup ausgeführt, welches das Programm installiert. Bei der ersten Benutzung muss man dann noch Benutzername, Name der Datenbank und das Passwort angeben und kann das Tool dann benutzen.

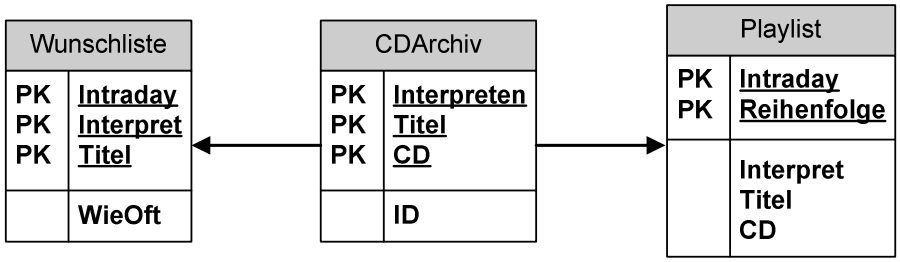
A-5 JFreeChart

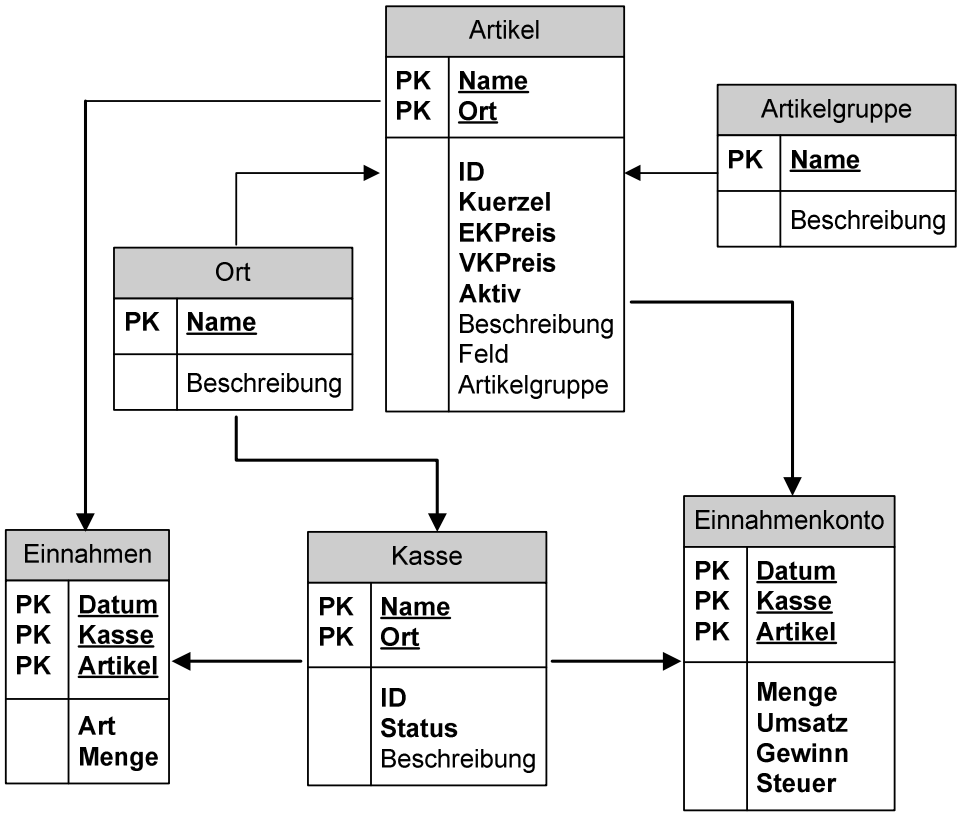
Zunächst lädt man sich die class library unter http://www.jfree.org/jfreechart/ aus dem Internet. Nachdem man die class library entpackt hat, kopiert man diese in das Verzeichnis [JavaHome]\lib\ext.

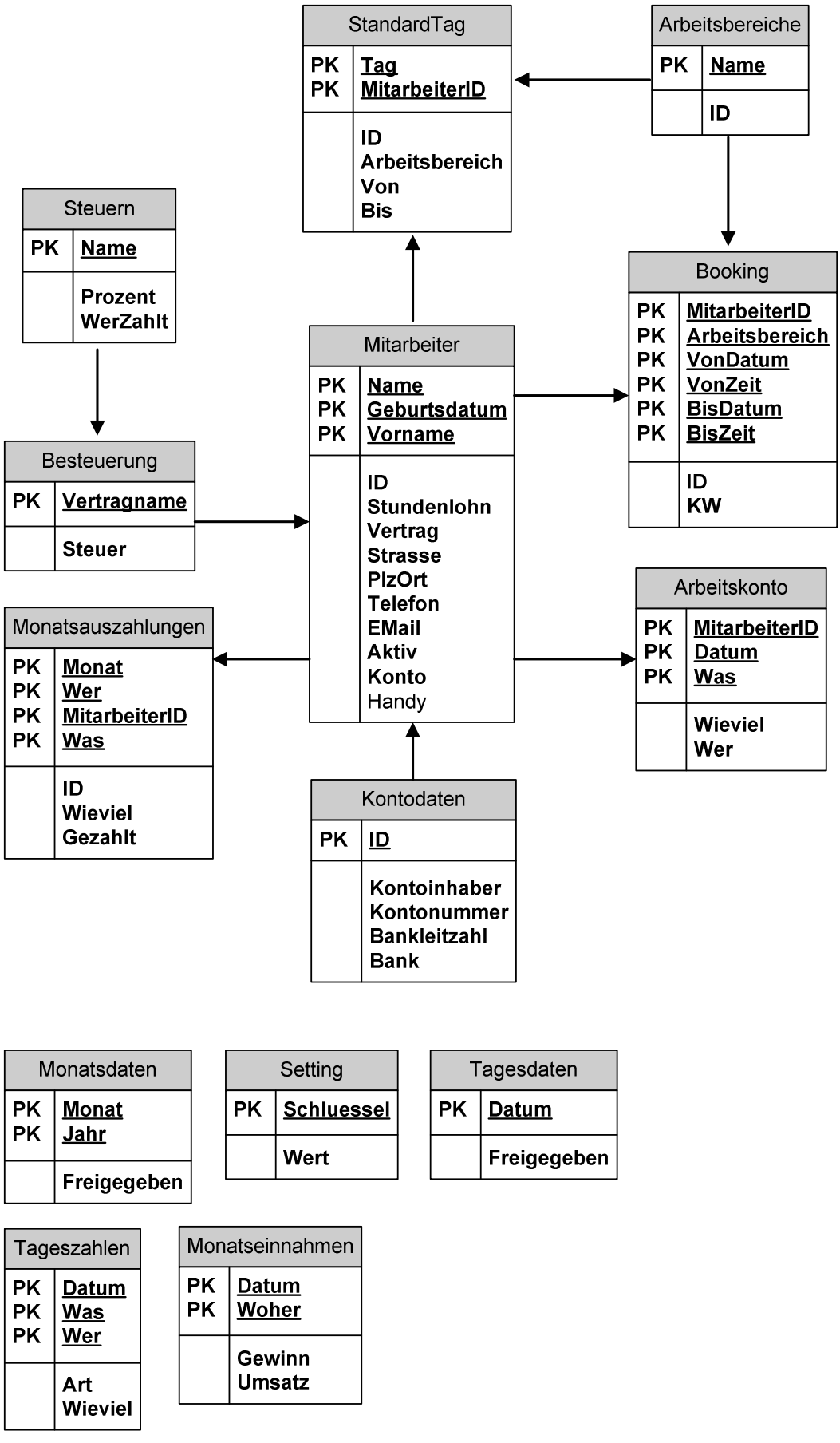
A-6 Tomcat

Nachdem man Tomcat unter http://jakarta.apache.org/tomcat/ herunter geladen hat, führt man die Setup-Datei aus. Die Installation von Tomcat ist in der aktuellen Version so gestaltet, dass Tomcat nach der Standardinstallation schon lauffähig und nutzbar ist. Um nun die Applikation zu installieren, wird die Tomcat-Administrations-Seite unter http://localhost:8080/admin/ nach dem Start von Tomcat aufgerufen. Auf dieser Seite kann man nun einen neue Applikation unter dem Namen Kopernikus anlegen. Nachdem diese Applikation angelegt ist, wird die Applikation einfach von der CD in das nun im Tomcat/webapps vorhanderene Verzeichnis kopiert. Nach einem Neustart von Tomcat ist die Installation abgeschlossen.

Anhang B ER-Diagramme

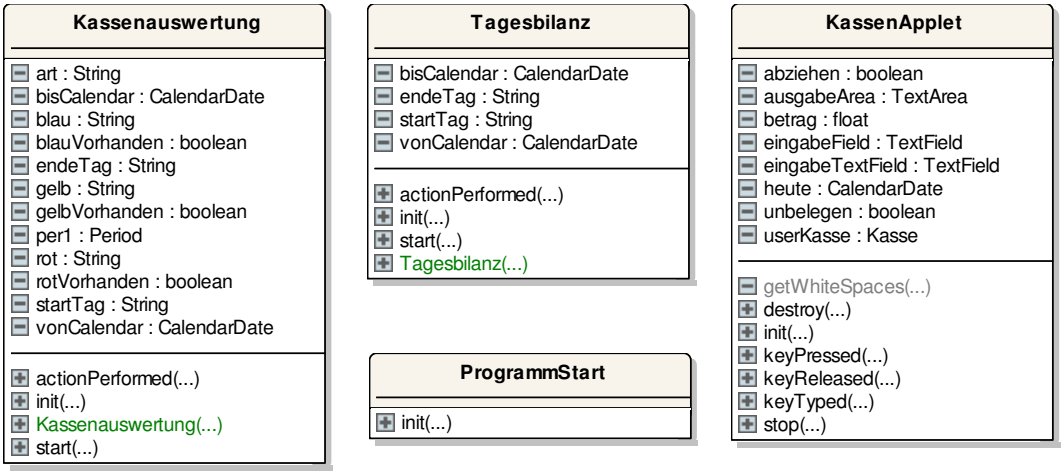




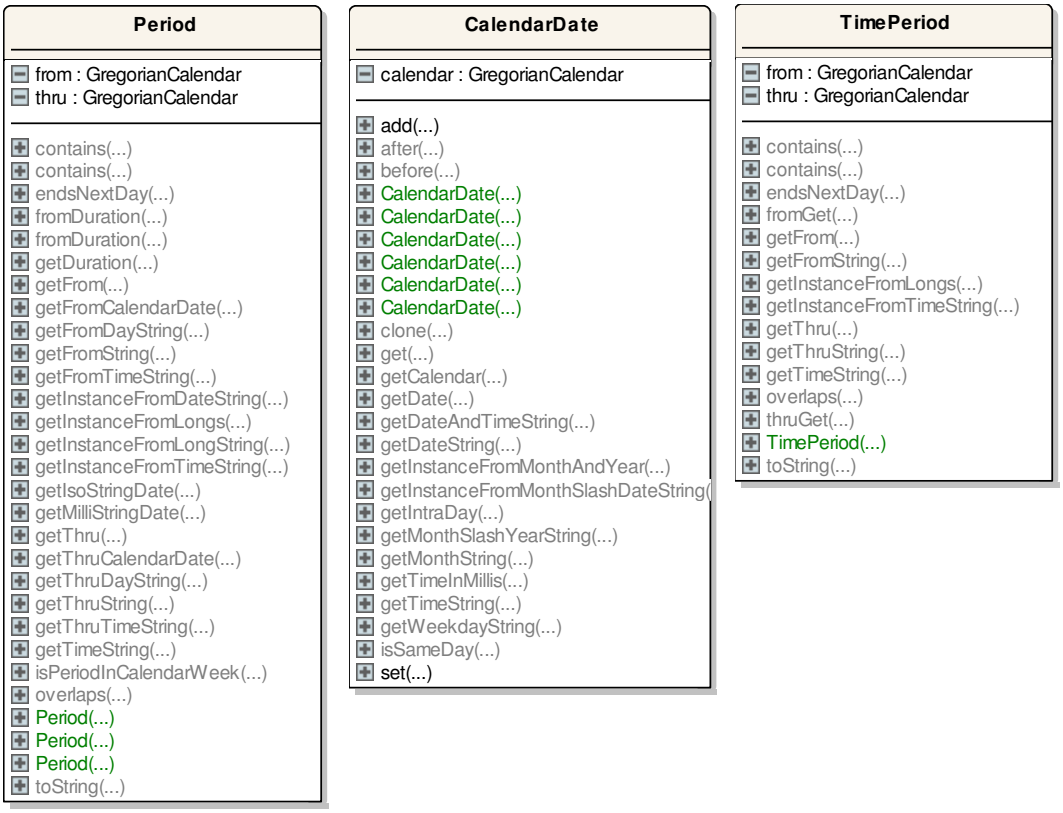

StandardTag
PK Tag
PK MitarbeiterID
ID
Arbeitsbereich
Von
Bis
Arbeitsbereiche
PK Name
ID
Steuern
PK Name
Prozent
WerZahlt
Booking
PK MitarbeiterID
PK Arbeitsbereich
PK VonDatum
PK VonZeit
PK BisDatum
PK BisZeit
ID
KW
Mitarbeiter
PK Name
PK Geburtsdatum
PK Vorname
ID
Stundenlohn
Vertrag
Strasse
PlzOrt
Telefon
EMail
Aktiv
Konto
Handy
Besteuerung
PK Vertragname
Steuer
Arbeitskonto
PK MitarbeiterID
PK Datum
PK Was
Wieviel
Wer
Monatsauszahlungen
PK Monat
PK Wer
PK MitarbeiterID
PK Was
ID
Wieviel
Gezahlt
Kontodaten
PK ID
Kontoinhaber
Kontonummer
Bankleitzahl
Bank
Monatsdaten
PK Monat
PK Jahr
Freigegeben
Setting
PK Schluessel
Wert
Tagesdaten
PK Datum
Freigegeben
Tageszahlen
PK Datum
PK Was
PK Wer
Art
Wieviel
Monatseinnahmen
PK Datum
PK Woher
Gewinn
Umsatz

Anhang C UML-Diagramme

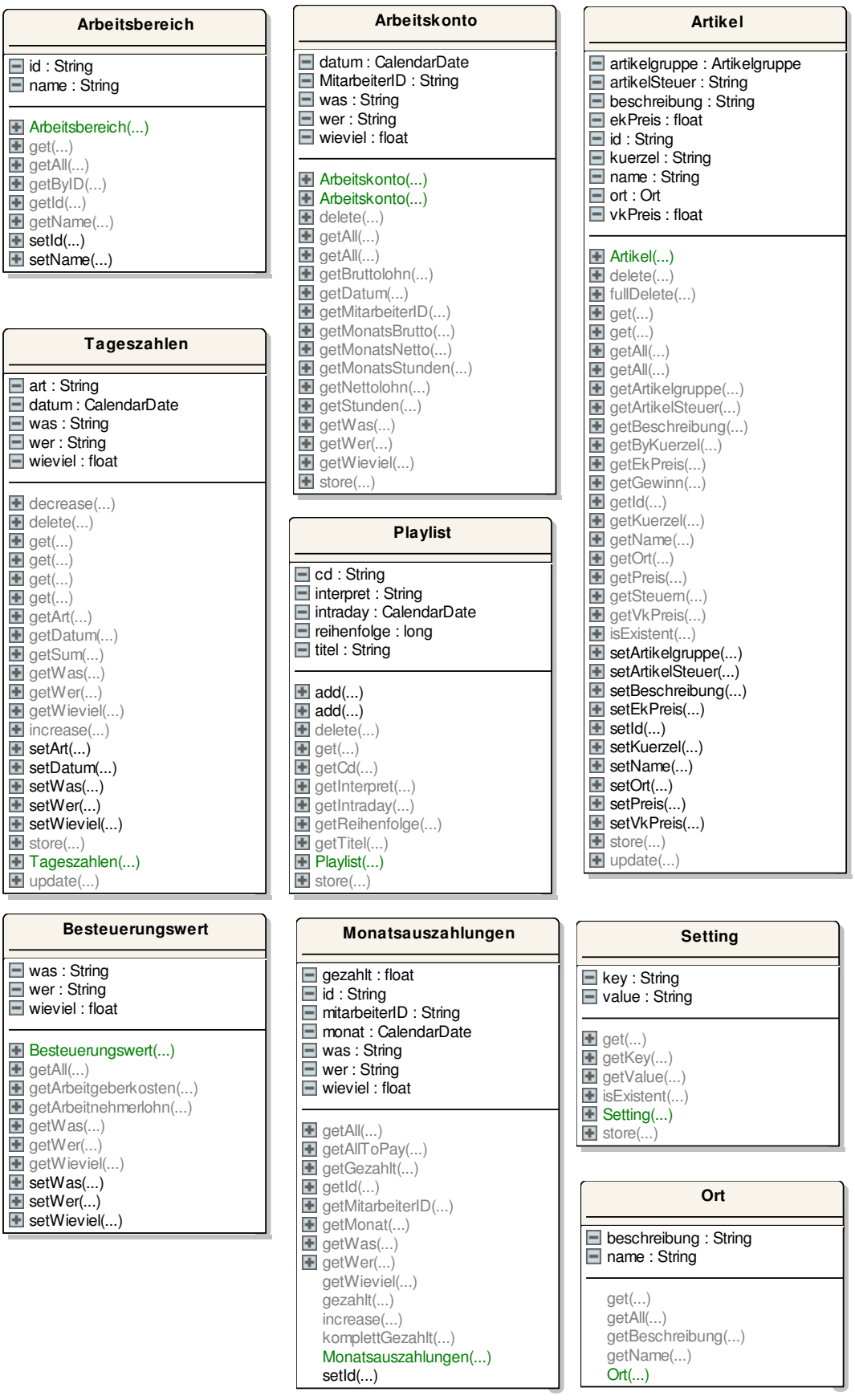
default



de.hoepp.date

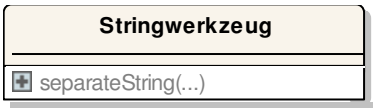


de.hoepp.kopernikus

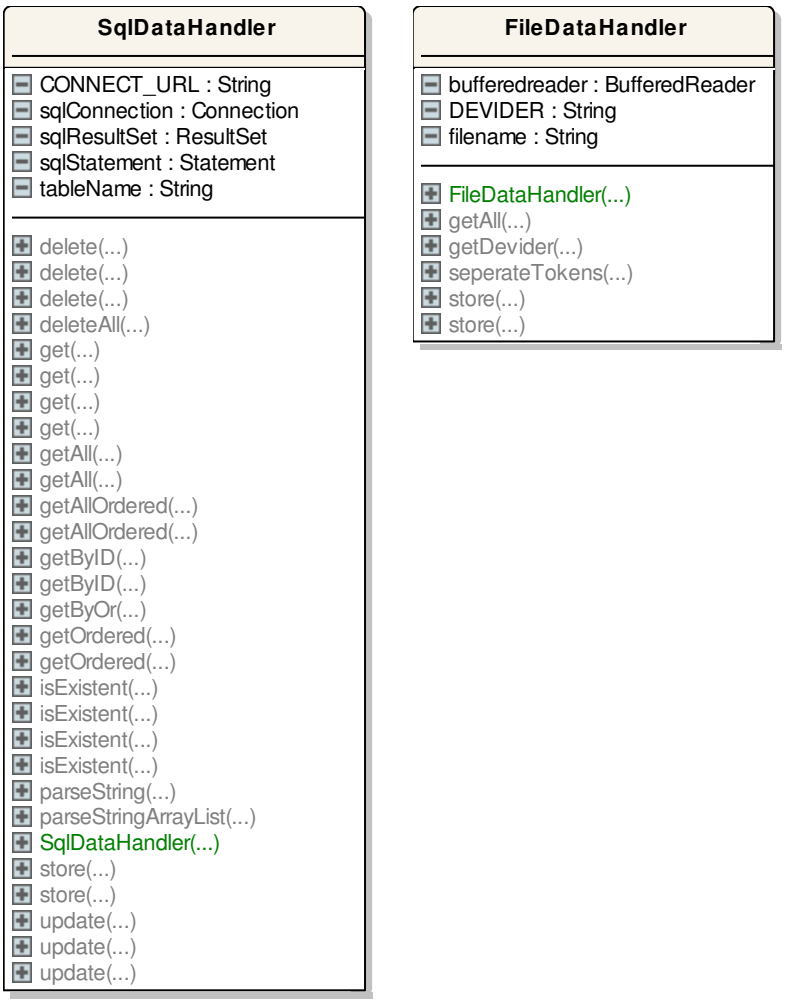



Mitarbeiter
eMail : String
gebDatum : CalendarDate
handy : String
id : String
kontodaten : Kontodaten
name : String
plzOrt : String
strasse : String
stundenlohn : String
telefon : String
vertrag : Vertrag
vorname : String
delete(...)
delete(...)
fullDelete(...)
get(...)
get(...)
getAll(...)
getAllOrdered(...)
getAllOrdered(...)
getEMail(...)
getGebDatum(...)
getHandy(...)
getId(...)
getKontodaten(...)
getName(...)
getPlzOrt(...)
getStrasse(...)
getStundenlohn(...)
getTelefon(...)
getVertrag(...)
getVorname(...)
isExistent(...)
Mitarbeiter(...)
setEMail(...)
setGebDatum(...)
setHandy(...)
setId(...)
setName(...)
setPlzOrt(...)
setStrasse(...)
setStundenlohn(...)
setTelefon(...)
setVertrag(...)
setVorname(...)
store(...)
update(...)
Tagesdaten
isDayReleased(...)
preselect(...)
releaseDay(...)
unreleaseDay(...)
Steuer
name : String
prozent : String
werZahlt : String
getName(...)
getProzent(...)
getSteuer(...)
getWerZahlt(...)
setName(...)
setProzent(...)
setWerZahlt(...)
Steuer(...)
Kasse
bezeichnung : String
id : String
name : String
ort : Ort
reserved : boolean
status : String
decrease(...)
getAllFree(...)
getBezeichnung(...)
getByID(...)
getID(...)
getName(...)
getOrt(...)
getStatus(...)
increase(...)
Kasse(...)
reserve(...)
setBezeichnung(...)
setID(...)
setName(...)
setOrt(...)
setStatus(...)
unreserve(...)
Besteuerung
vertragName : String
vertragSteuer : Steuer
Besteuerung(...)
delete(...)
getAll(...)
getAll(...)
getAllSteuern(...)
getVertragName(...)
getVertragSteuer(...)
setVertragName(...)
setVertragSteuer(...)
store(...)
Monatseinnahmen
datum : CalendarDate
gewinn : float
umsatz : float
woher : String
getAll(...)
getDatum(...)
getGewinn(...)
getSums(...)
getUmsatz(...)
getWoher(...)
increase(...)
Monatseinnahmen(...)
Kontodaten
bank : String
bankleitzahl : int
id : int
kontoinhaber : String
kontonummer : int
get(...)
getBank(...)
getBankleitzahl(...)
getId(...)
getKontoinhaber(...)
getKontonummer(...)
Kontodaten(...)
setBank(...)
setBankleitzahl(...)
setId(...)
setKontoinhaber(...)
setKontonummer(...)
store(...)
update(...)
Wunschliste
interpret : String
intraday : CalendarDate
reihenfolge : long
titel : String
wieOft : int
add(...)
delete(...)
get(...)
get(...)
getAll(...)
getInterpret(...)
getIntraday(...)
getReihenfolge(...)
getTitel(...)
getWieOft(...)
setInterpret(...)
setIntraday(...)
setReihenfolge(...)
setTitel(...)
setWieOft(...)
Wunschliste(...)
Einnahmenkonto
artikel : Artikel
datum : CalendarDate
gewinn : float
kasse : String
menge : int
ort : Ort
steuer : float
umsatz : float
delete(...)
Einnahmenkonto(...)
get(...)
getAll(...)
getArtikel(...)
getDatum(...)
getGewinn(...)
getKasse(...)
getMenge(...)
getOrt(...)
getSteuer(...)
getSums(...)
getUmsatz(...)
increase(...)
Booking
bereichsID : String
id : String
mitarbeiterID : String
zeitraum : Period
zeitraumString : String
Booking(...)
delete(...)
delete(...)
getAll(...)
getAll(...)
getAll(...)
getAllOrdered(...)
getBereichsId(...)
getId(...)
getMitarbeiterID(...)
getZeitraum(...)
getZeitraumString(...)
setId(...)
store(...)

de.hoepp.util

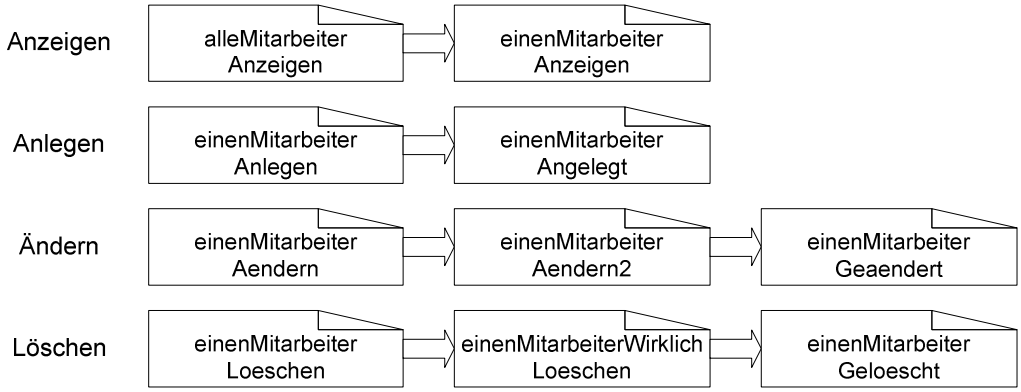


de.hoepp.dataHandler

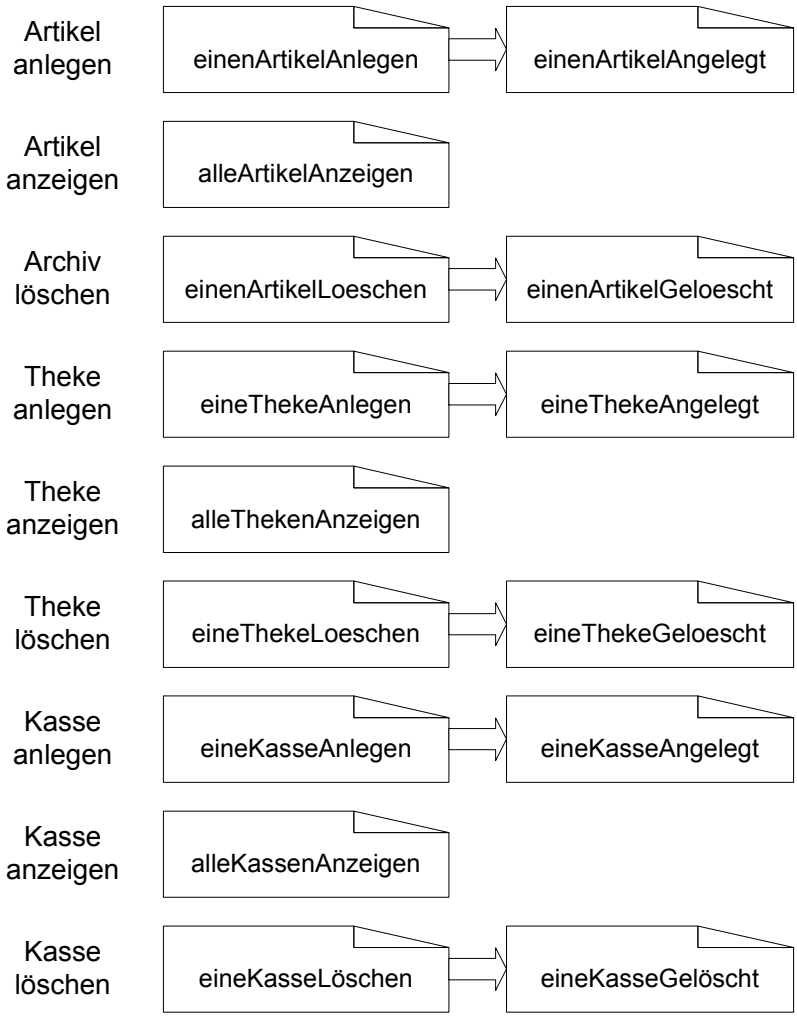


Anhang D Seitenverlauf

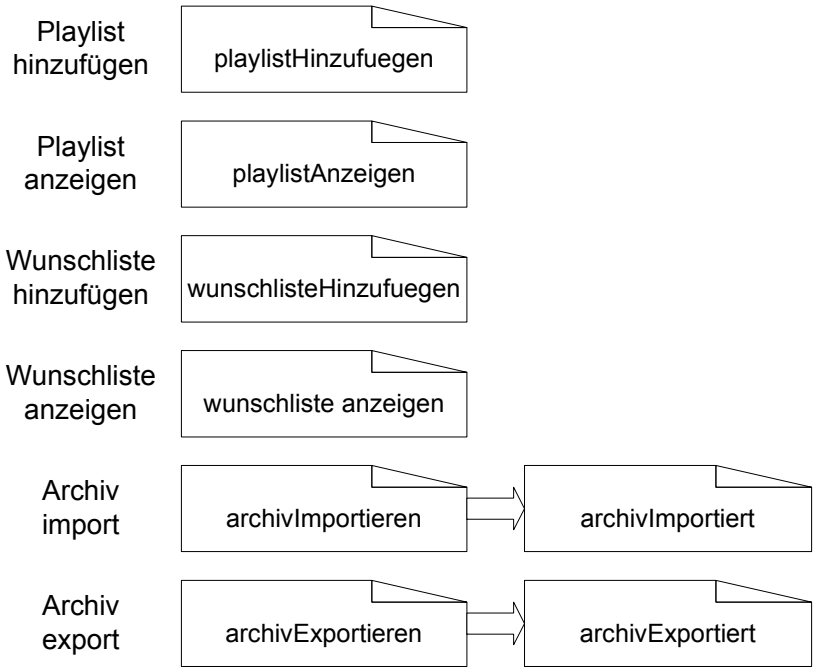
Modul Mitarbeiterverwaltung



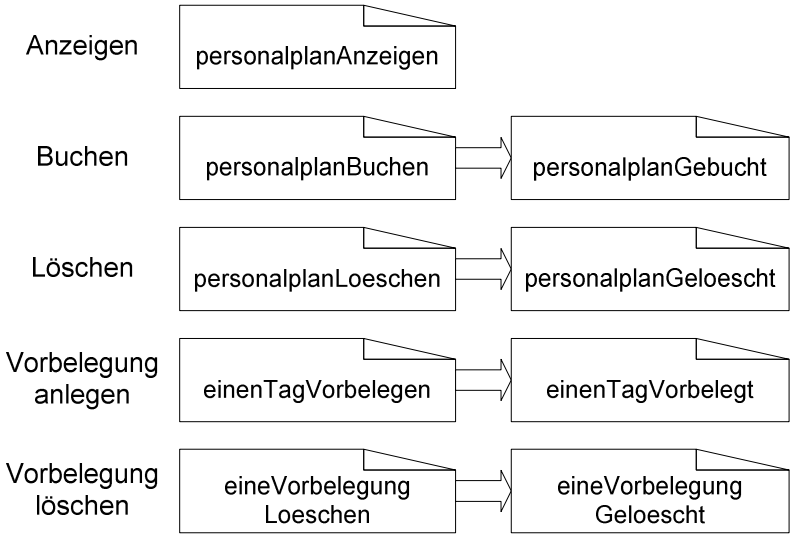
Modul Administration



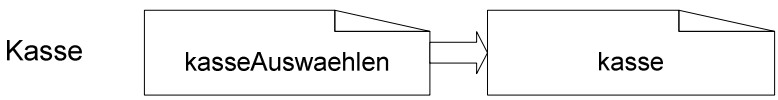
Modul Deejaying



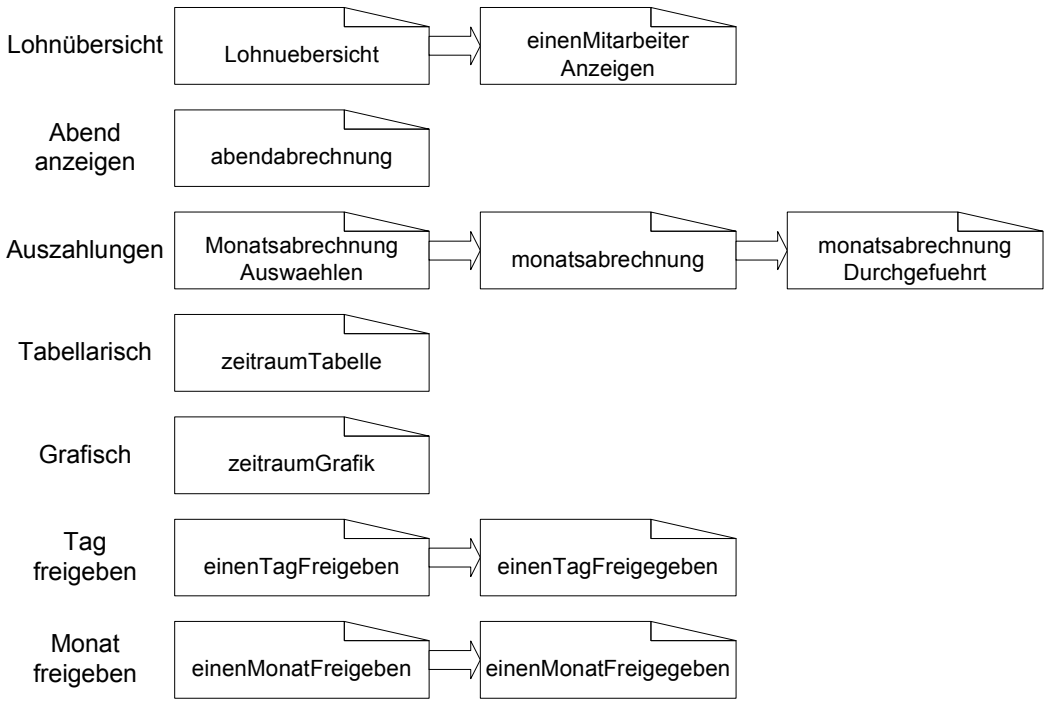
Modul Mitarbeitermanagement



Modul Kassensystem



Modul Abrechnung



Anhang E Reguläre Ausdrücke

Hier eine kleiner Auszug aus den Regeln für reguläre Ausdrücke aus dem Kapitel CGI/Perl-Kapitel in [SelfHtml]

Reguläre Ausdrücke für einzelne Zeichen

Sie können in einer Zeichenkette

- nach einem bestimmten Zeichen suchen
- nach mehreren bestimmten Zeichen suchen
- nach einem Zeichen aus einem bestimmten Bereich suchen

Solche Ausdrücke eignen sich weniger zur Mustererkennung als dazu, um herauszufinden, ob in einem Wert bestimmte Zeichen, z.B. unerlaubte Zeichen vorkommen.

Regulärer Ausdruck (Beispiel)	passt auf eine Zeichenkette, die (mindestens)
`/a/`	ein 'a' enthält
`/[ab]/`	ein 'a' oder ein 'b' enthält
`/[A-Z]/`	einen Großbuchstaben enthält (passt nicht auf Umlaute)
`/[0-9]/`	eine Ziffer enthält
`/\d/`	eine Ziffer enthält - genau wie (4.)
`/\D/`	ein Zeichen enthält, das keine Ziffer ist
`/[-\d]/`	eine Ziffer oder ein Minuszeichen enthält
`/[\[\]]/`	eine eckige Klammer enthält
`/[a-zA-Z0-9_]/`	eins der Zeichen vom Typ Buchstabe (ohne Umlaute), vom Typ Ziffer oder einen Unterstrich enthält
`/\w/`	eins der Zeichen vom Typ Buchstabe, vom Typ Ziffer oder einen Unterstrich enthält - (fast) genau wie (9.); ob Umlaute erkannt werden können, hängt von der Systemkonfiguration ab

/\W/	ein Zeichen enthält, was weder Buchstabe noch Ziffer noch Unterstrich ist; ob Umlaute ausgeschlossen werden können, hängt von der Systemkonfiguration ab
/\r/	ein Steuerzeichen für den Wagenrücklauf enthält
/\n/	ein Steuerzeichen für den Zeilenvorschub enthält
/\t/	ein Steuerzeichen für den Tabulator enthält
/\f/	ein Steuerzeichen für den Seitenvorschub enthält
/\s/	ein Leerzeichen oder ein Steuerzeichen aus (12.-15.) enthält
/\S/	ein Zeichen enthält, das kein Leerzeichen oder Steuerzeichen aus (12.-15.) ist
/[^äöüÄÖÜ]/	ein Zeichen enthält, was kein deutscher Umlaut (in der entsprechenden Zeichencodierung) ist
/[^a-zA-Z]/	ein Zeichen enthält, welches kein Buchstabe ist (ohne Umlaute)

Reguläre Ausdrücke für Zeichenketten

Sie können

- nach einer bestimmten Zeichenkette suchen
- nach einer Zeichenketten mit Gruppierungsoperatoren (Platzhalter, Wildcards) suchen
- nach Zeichenketten am Anfang oder Ende eines Wortes suchen
- nach Zeichenketten am Anfang oder Ende einer Zeile suchen

Diese Art von regulären Ausdrücken ist dazu gedacht, um etwa in einem Wert nach dem Vorkommen eines bestimmten Wortes, einer beliebigen Teilzeichenkette oder nach einem Muster zu suchen.

Regulärer Ausdruck (Beispiel)	Wirkung
`/aus/`	passt auf 'aus' - auch in 'Haus' oder 'Mausi'
`/aus?/`	passt auf 'aus' usw. - aber auch 'au' und 'auf'
`/a./`	passt auf 'ab' und 'an' (ein beliebiges Zeichen hinter 'a', außer `\n`)
`/a+/`	passt auf 'a' und 'aa' und 'aaaaa' (ein oder beliebig viele 'a')
`/a*/`	passt auf 'a' und 'aa' und 'aaaaa' und 'b' (kein oder beliebig viele 'a')
`/Ha.s/`	passt auf 'Haus' und 'Hans' aber nicht 'Hannes'
`/Ha.+s/`	passt auf 'Haus' und 'Hans' und 'Hannes' (ein oder beliebig viele beliebige Zeichen, außer `\n`)
`/Ha.s/`	passt auf 'Haus' und 'Hans' aber nicht 'Hase'
`/Ha.?s/`	passt auf 'Haus' und 'Hans' und 'Hase'
`/x{10,20}/`	passt auf zwischen 10 und 20 'x' in Folge
`/x{10,}/`	passt auf 10 und mehr 'x' in Folge
`/x.{2}y/`	passt auf 'xxxy' oder 'xaby' usw. (zwei beliebige Zeichen zwischen 'x' und 'y', außer `\n`)
`/Hans\b/`	passt auf 'Hans' aber nicht 'Hansel' (Wortgrenze)
`/\baus/`	passt auf 'aus' oder 'aussen' aber nicht 'Haus' (Wortgrenze)
`/\baus\b/`	passt auf 'aus' aber nicht 'Haus' und auch nicht 'aussen' (Wortgrenze)
`/\baus\B/`	passt auf 'aussen' aber nicht 'aus' und auch nicht 'Haus' (Wortgrenze und "negative"

	Wortgrenze)
`/^Hans/`	passt auf 'Hans' nur am Anfang des zu durchsuchenden Bereichs
`/Hans$/`	passt auf 'Hans' nur am Ende des zu durchsuchenden Bereichs
`/^\s*$/`	passt auf Zeilen, die nur aus Leerzeichen und anderen Leerraumzeichen bestehen oder leer sind

Anhang F Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Seiten, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

_____

Köln, den 9.8.2004

Stefan Höpp